(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 November 2008 (27.11.2008)

PCT

(10) International Publication Number
WO 2008/141427 A1

(51) International Patent Classification:
H04L 12/16 (2006.01) *G06Q 30/00* (2006.01)
G06F 17/00 (2006.01) *H04Q 7/22* (2006.01)

(21) International Application Number:
PCT/CA2008/000909

(22) International Filing Date: 12 May 2008 (12.05.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
607924,503 17 May 2007 (17.05.2007) US

(71) Applicant *(for all designated States except US)*: **FAT FREE MOBILE INC.** [CA/CA]; 3872 Swiftdale Drive, Mississauga, Ontario L5M 6M2 (CA).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **KIM, Sang-Heun** [CA/CA]; 2610-33 Elm Drive West, Mississauga, Ontario L5B 4M2 (CA). **STINSON, Charles, Laurence** [CA/CA]; 3872 Siftdale Drive, Mississauga, Ontario L5M 6M2 (CA).

(74) Agent: **Gowling Lafleur Henderson LLP**; Suite 1600, 1 First Canadian Place, 100 King Street West, Toronto, Ontario M5X 1G5 (CA).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(54) Title: METHOD AND SYSTEM FOR AUTOMATICALLY GENERATING WEB PAGE TRANSCODING INSTRUCTIONS

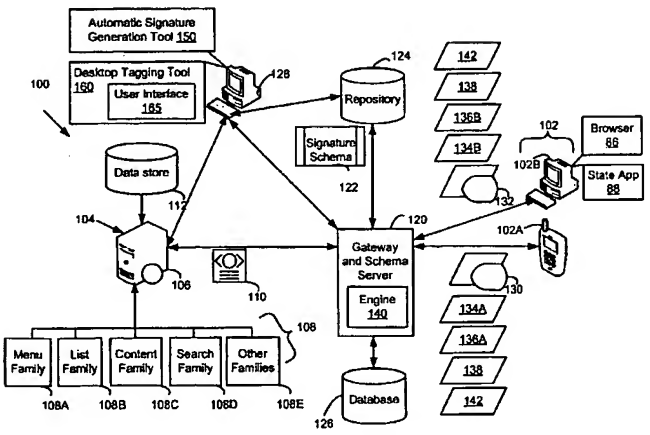

Figure 1

(57) Abstract: A system and method are provided for generating transcoding instructions to identify and extract a subset of data from a web page. Input describing the subset of data is received where the input describes one or more data fields and, for each data field, respective field values from at least two sample web pages of a web page family for the web site. For each field, respective web page code defining the respective field values may be compared for commonality to find a matching pattern with which to locate the respective field values. The matching pattern comprises a signature for the data field. Transcoding instructions are defined using the matching pattern to locate and extract field values within web pages of the web page family. The subset of data may be expressed in a target format to transcode the web page for particular client machines (e.g. a wireless mobile device).

WO 2008/141427 A1

METHOD AND SYSTEM FOR AUTOMATICALLY GENERATING WEB PAGE TRANSCODING INSTRUCTIONS

CROSS-REFERENCE

[0001] This application claims the benefit of the prior filing of U.S. Provisional Patent Application Serial No. 60/924503 filed May 17, 2007, the disclosure of which is incorporated herein by reference.

COPYRIGHT

FIELD

[0003] The present application relates generally to telecommunications and more particularly to a method and system for automatically generating web page transcoding instructions.

BACKGROUND

[0004] Web sites host and provide information using web pages that are communicated electronically via a telecommunications network. Accessing this information by some client computing devices can be challenging. Computing devices are becoming smaller and increasingly utilize wireless connectivity. Examples of such computing devices include portable computing devices that include wireless network browsing capability as well as telephony and personal information management capabilities.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] Figure 1 is schematic representation of a system for content navigation.

[0006] Figure 2 is a schematic representation of a wireless communication device from Figure 1.

[0007] Figure 3 illustrates a flow of interactions among components of the system of Figure 1.

[0008] Figure 4 is a schematic representation of a system for content navigation in accordance with another embodiment.

[0009] Figure 5 illustrates a flow of interactions among components of the system of Figure 4.

[0010] Figure 6 illustrates an exemplary operations of an automatic signature creation tool of the system of Figure 1.

[0011] Figure 7 illustrates an exemplary view of a user interface of a desktop tagging tool for indentifying a subset of data on a web page in accordance with an embodiment.

[0012] Figures 8A–8D and 9A-9D respectively illustrate representative web pages rendered on a first browser window and portions of said representative web pages transcoded and rendered on a second browser window in accordance with an embodiment.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0013] The smaller size of most wireless mobile client devices necessarily limits their display capabilities. Furthermore the wireless connections to such devices typically have less or more expensive bandwidth than corresponding wired connections. The Wireless Application Protocol ("WAP") was designed to address such issues, but WAP can still provide a very unsatisfactory experience or even completely ineffective experience, particularly where the small client device needs to effect a connection with web sites that host web pages that are directed to traditional full desktop browsers.

[0014] A system and method are provided for generating transcoding instructions to identify and extract a subset of data from a web page. Input describing the subset of data is received where the input describes one or more data fields and, for each data field, respective field values from at least two sample web pages of a web page family for the web site. For each field, respective web page code defining the respective field values may be compared for commonality to find a matching pattern with which to locate the respective field values. The matching pattern may define a signature for the data field. Transcoding

instructions are defined using the matching pattern to locate and extract field values within web pages of the same web page family. The subset of data may be expressed in a target format to transcode the web page for particular client machines (e.g. wireless mobile device).

[0015] In accordance with an aspect there is provided a method of automatically generating transcoding instructions to locate and extract a subset of data from a selected web page of a web site. The method comprises receiving an input describing the subset of data, said input comprising one or more data fields and, for each data field, respective field values from at least two sample web pages of a web page family for the web site; and for each data field: comparing respective web page code defining the respective field values for commonality to find a matching pattern with which to locate the respective field values, said matching pattern comprising a signature for the data field; and defining the transcoding instructions in accordance with the matching pattern to locate and extract field values for the data field within web pages of the web page family.

[0016] Comparing respective web page code defining the respective field values may comprise locating the respective field values in the respective web page code. Comparing respective web page code defining the respective field values may further comprise locating object tags within the web page code. The method may further comprising constructing a programmatic data structure representing a hierarchy of object tags within the web page code and reviewing the hierarchy to determine the commonality.

[0017] Comparing respective web page code may comprise performing pattern recognition to define a common pattern within the web page code with which to locate the respective field values.

[0018] The web page code may comprise markup language in plain text. Each signature may comprise characters selected from the plain text of the web page code.

[0019] The method may further comprising automatically defining the input in accordance with a tagging tool that identifies the respective field values from the sample web pages. The web site may comprise an e-commerce web site for making a purchase.

[0020] The method may further comprise defining transcoding instructions to express the extracted subset of data in a target format thereby to transcode the web page.

[0021] In accordance with another aspect, there is provided a system for automatically generating transcoding instructions to locate and extract a subset of data from a selected web page of a web site. The system comprises a processor and memory coupled thereto, said memory storing instructions and data to configure the processor for: receiving an input describing the subset of data, said input comprising one or more data fields and, for each data field, respective field values from at least two sample web pages of a web page family for the web site; and for each data field: comparing respective web page code defining the respective field values for commonality to find a matching pattern with which to locate the respective field values, said matching pattern comprising a signature for the data field; and defining the transcoding instructions in accordance with the matching pattern to locate and extract field values for the data field within web pages of the web page family.

[0022] Yet another aspect provides a computer program product for automatically generating transcoding instructions to locate and extract a subset of data from a selected web page of a web site, the computer program product storing computer readable instructions which when executed by a computer processor configure the processor to: A computer program product for automatically generating transcoding instructions to locate and extract a subset of data from a selected web page of a web site, the computer program product storing computer readable instructions which when executed by a computer processor configure the processor to: receive an input describing the subset of data, said input comprising one or more data fields and, for each data field, respective field values from at least two sample web pages of a web page family for the web site; and for each data field: compare respective web page code defining the respective field values for commonality to find a matching pattern with which to locate the respective field values, said matching pattern comprising a signature for the data field; and define the transcoding instructions in accordance with the matching pattern to locate and extract field values for the data field within web pages of the web page family.

[0023] Referring now to Figure 1, there is illustrated a system 100 for content navigation

via a telecommunications network. In a present embodiment system 100 comprises a plurality of client computing devices in the form of client machines 102A and 102B (collectively 102), a web site server 106 hosting a web site 104 and a gateway and schema server 120. Devices 102 are respectively coupled to communicate with gateway and schema server 120 to obtain web pages (e.g. 110) transcoded from web site 104.

[0024] In the present embodiment, a web server 106 serves web pages (e.g. 110) which comprise web site 104. The web pages are defined from a plurality of web page family templates 108A-108D (collectively 108) and web page content (described further herein below) from data store 112. For ease within the present embodiment, only a single web site 104 is shown coupled via gateway and schema server 120; however, in another embodiment a plurality of different web sites may be so coupled. In the present embodiment of system 100, gateway and schema server 120 is coupled to a schema repository 124 from which to obtain a signature schema 122 for a particular web site. Signature schema documents (e.g.122) provide instructions and data with which an engine 140 of server 120 can extract data from web pages (e.g. 110) and transcode same to a target format to provide transcoded web page data (e.g. 130 and 132) to the respective requesting client machines 102A and 102B as described more fully below. Gateway and schema server 120 may also be coupled to a database 126 for retrieving/storing data extracted from web sites in accordance with its operations. The database 126 may be a relational database storing extracted data from web sites in relation to the defined signature schema. The stored data can be accessed by a Structured Query Language (SQL). Signature schemas for respective web sites may be defined (e.g. coded) using a computing device 128 as described herein below.

[0025] Representative client machines 102 include any type of computing or electronic device that can be used to communicate and interact with content available via web sites. Each of the client machines 102 may be operated by a respective user U (not shown). Interaction with a particular user includes presenting information on a client machine (e.g. by rendering on a display screen) as well as receiving input at a client machine (e.g. such as via a keyboard for transmitting to a web site). In the present embodiment, client

machine 102A comprises a mobile electronic device with the combined functionality of a personal digital assistant, cell phone, email paging device, and a web-browser. Such a mobile electronic device may comprise a keyboard (or other input device(s)), a display screen, a speaker, (and other output device(s) (e.g. LEDs)) and a chassis for housing such components. The chassis may further house one or more central processing units, volatile memory (e.g. random access memory), persistent memory (e.g. Flash read only memory) and network interfaces to allow client machine 102A to communicate over the telecommunication network.

[0026] Referring now to Figure 2, a schematic block diagram shows an exemplary client machine 102A in greater detail. It should be emphasized that the structure in Figure 2 is purely exemplary, and contemplates a device that may be used for both wireless voice (e.g. telephony) and wireless data (e.g. email, web browsing, text) communications. Client machine 102A includes a plurality of input devices which in a present embodiment includes a keyboard and, typically, additional input buttons, collectively 200, an optional pointing device 202 (e.g. a trackball or trackwheel) and a microphone 204. Other input devices, such as a touch screen, and camera lens are also contemplated. Input from keyboard/buttons 200, pointing device 202 and microphone 204 may be received at a processor 208. Processor 208 may be further operatively coupled with a non-volatile storage unit 212 (e.g. read only memory ("ROM"), Erasable Electronic Programmable Read Only Memory ("EEPROM"), or Flash Memory) and a volatile storage unit 216 (e.g. random access memory ("RAM"), speaker 220, display screen 224 and one or more lights (LEDs 222). Processor 208 may be operatively coupled for network communications via a subsystem 226. Wireless communications are effective via at least one radio (e.g. 228) such as for Wi-Fi or cellular wireless communications. Client machine 102A also may be configured for wired communications such as via a USB or other port and for short range wireless communications such as via a Bluetooth® radio (all not shown).

[0027] Programming instructions that implement the functional teachings of client machine 102A as described herein are typically maintained, persistently, in non-volatile storage unit

212 and used by processor 208 which makes appropriate utilization of volatile storage 216 during the execution of such programming instructions. Of particular note is that non-volatile storage unit 212 persistently maintains a web browser application 86 and, in the present embodiment, a native menu application 82, each of which can be executed on processor 208 making use of volatile storage 216 as appropriate. An operating system and various other applications (not shown) are maintained in non-volatile storage unit 212 according to the desired configuration and functioning of client machine 102A, one specific non-limiting example of which is a contact manager application (also known as an address book, not shown) which stores a list of contacts, addresses and phone numbers of interest to user U and allows user U to view, update, and delete those contacts, as well as providing user U an option to initiate telecommunications (e.g. telephone, email, instant message (IM), short message service (SMS)) directly from that contact manager application.

[0028] Native menu application 82 may be configured to provide menu choices to user U according to the particular application (or other context) that is being accessed. By way of example, while user U is activating the contact manager application, user U can activate menu application 82 to access a plurality of menu choices available that are respective to contact manager application 90. For example, menu choices may include options to invoke other applications (e.g. a mapping application to map a contact's address) or communication functions (e.g. call, SMS, IM, email, etc.) on the client machine 102A for a particular contact. Menu application 82 may be associated to a particular input button (e.g. one of buttons 200) and invoked to provide a contextual menu comprised of a plurality of menu choices that are reflective of the context in which the button 200 was selected. Note that the options in a contextual menu are stored within non-volatile storage 212 as being specifically associated with a respective application. Menu application 82 may be therefore configured to generate a plurality of different contextual menus that are reflective of the particular context in which the menu application 82 is invoked. For example, in an email application where an email is being composed, invoking menu application 82 would generate a contextual menu that included the options of sending the email, cancelling the

email, adding addresses to the email, adding attachments, and the like. The contents for such a contextual menu would also be maintained in non-volatile storage 212. Other examples of contextual menus will occur to those of ordinary skill in the art.

[0029] As noted, gateway and schema server 120 applies a signature schema to transcode a web page and provide transcoded data to a requesting client machine 102. Signature schema 122 may be configured to transcode navigational features of a web site 104 to provide menu options to menu application 82 for use when browsing the web site 104 with browser 86. The signature schema may further transcode web site content for presentation by the browser 86.

[0030] Figures 8A–8D and 9A-9D respectively illustrate representative web pages rendered on a first browser window and portions of a subset of data from said representative web pages transcoded and rendered on a second browser window in accordance with an embodiment. Figure 8A illustrates a representative home web page 660A of an e-commerce web site (e.g. 104) in a browser window 650. Window 650 is illustrative of a rendering to a large size display device (e.g. desktop monitor). Web page 660A comprises, among other things, a menu portion 652 and a primary content display portion 654, in the example, showing various advertisements 655 for products. Figure 9A illustrates the menu portion 652 extracted and transcoded and rendered as a web page on a second browser window 750. Window 750 is illustrative of a rendering to a small size display device (e.g. of a wireless mobile device). In addition to transcoding as a web page, menu portion 652 may be transcoded for menu application 82 e.g. for invocation when browsing the site 104 as referenced further herein.

[0031] Figure 8B illustrates an exemplary product web page 660B in window 650 showing various product data (collectively 666) including image 666A, price 666, title 666C and description 666D data that is transcoded and shown in window 750 of Figure 9B. Also transcoded is the web page hierarchy list 668 showing where the page is on the web site.

[0032] Figure 8C illustrates an exemplary product list web page 660C in window 650

showing a list of products (collectively 670). A subset of the product data such as image 670A, price 670B, and title 670C is transcoded and shown in window 750 of Figure 9C. Note that multiple pages 672 may be provided for the list 670.

[0033] Figure 8D illustrates an exemplary account checkout web page 660D in window 650 showing a login form 680 for receiving account login and password, which form is transcoded and shown in window 750 of Figure 9D. Though not shown, other checkout pages (e.g. for payment or order confirmation, etc.), search pages, product and information pages may be similarly transcoded.

[0034] Returning now to Figure 1, web server 106 and gateway and schema server 120 (which can, if desired, be implemented on a single server) can be based on any commonly available server environments or platforms including a module that houses one or more central processing units, volatile memory (e.g. random access memory), persistent memory (e.g. hard disk devices) and network interfaces to allow servers 106 and 120 to communicate over the telecommunications network. Web server 106 hosts software applications comprising instructions and data for generating and serving web pages dynamically from the template families 108 and current informational content therefore from data store 112. Load balancing, security/firewall, billing, account and other applications may also be present.

[0035] Gateway and schema server 120 hosts software applications comprising instructions and data for proxying requests and responses between the client machines 102 and web site 104. In addition to software for maintaining HTTP communications, performing requests, maintaining sessions, handling cookies, etc., engine 140 may be implemented in software to apply the signature schemas to web pages from web sites. There may be provided an interpreter that interprets the signature schema document and applies the actions against the web page code (as an ASCII (plain text) document) to extract the subset of data to produce a result set. A renderer may be provided to express the subset of data result set (i.e. transcode to a target format such as cHTML (Compact HTML) for a mobile device browser) for transmitting to the client machines also in

accordance with the signature schema. A cache feature may also be provided for storing/retrieving data from database 126. Caching may comprise storing web pages from the web site as well as extracted data from which to build a relational database of object and elements and their relationships. The gateway and schema server (or a separate server (not shown)) may host a web site engine to provide content extracted from the relational database (e.g. stored web site data) to the client machines 102.

[0036] Devices 102, schema server 120 and web site 104 are coupled via a telecommunication network (not shown) typically comprising a plurality of interconnected networks that may include wired and (at least for device 102A) wireless networks. It should now be understood that the nature of the network is not particularly limited and is, in general, based on any combination of architectures that will support interactions between client machines 102 and servers 106 and 120. In a present embodiment the network includes the Internet as well as appropriate gateways and backhauls.

[0037] More specifically, in the present embodiment, a wireless network for client machine 102A may be based on core mobile network infrastructure (e.g. Global System for Mobile communications ("GSM"), Code Division Multiple Access ("CDMA"), Enhanced Data rates for GSM Evolution ("EDGE"), Evolution Data-Optimized ("EV-DO"), High Speed Downlink Packet Access ("HSPDA"), Universal Mobile Telecommunications System ("UMTS"), etc.) or on wireless local area network ("WLAN") infrastructures such as the Institute for Electrical and Electronic Engineers ("IEEE") 802.11 Standard (and its variants) or Bluetooth or the like or hybrids thereof. In the present embodiment of system 100 it is contemplated that client machine 102B may be another type of client machine such as a PC (desktop or laptop) configured to include a full desktop computer or as a "thin-client". Typically such have larger display monitors/screens than portable machines like 102A. A wired network for system 100 and device 102B can be based on a T1, T3 or any other suitable wired connection.

[0038] As previously stated in relation to Figures 1 and 2, each of the client machines 102 is configured to interact with content available over the network, including web pages on

web site 104. In a present embodiment, client machines 102A and 102B may navigate for content using a browser application (e.g. 86). As will be explained further below, on client machine 102A, browser application 86 may be a mini-browser in the sense that it may be configured to render web pages on the relatively small display 224 of client machine 102A. Often, during such rendering, those pages are presented in a format that may be different from how those pages are rendered on a traditional desktop browser application (e.g. browser 86 of client machine 102B). Mini-browsers typically attempt to convey substantially the same information as if the web pages had been rendered on a full browser such as Internet Explorer®, Safari® or Firefox® on a traditional desktop or laptop computer like client machine 102B.

[0039] Figure 3 is a flowchart illustrating operations/interactions for transcoding a web page (e.g. 110) from web site 104 for client machine 102A, providing an example of the interaction among the gateway and schema server 120, client machine 102A and the web site 104. Client machine 102A makes a request 302 to server 120, acting as a proxy, for a specific web page (e.g. 110) from a web site having a specific domain (URL). The gateway and schema server engine 140 receives the request and makes a corresponding request 304 as a proxy to the web site's web server 106 for the specified page, receiving 308 the web page code (e.g. 110) into the engine's (140) memory. The web page code is treated as an ASCII (plain text) file. It typically does not include objects referenced by the code such as images, video, audio, further web pages, etc. that are typically subsequently retrieved and inserted at the time of rendering a web page by a browser.

[0040] The engine 140 (for example, in parallel or without waiting for a response from server 106) makes a request 306 to the signature repository 124 for the signature schema document 122 for the web site, which request may use the domain in the URL as an identifier for obtaining the document 122. The engine 140 receives 310 the schema. The engine 140 does not render the web page 110 per se but instead uses the instructions in the signature schema document 122 to extract the subset of data from the web page 110 for transcoding. In the present embodiment signature schema 122 is configured to

transcode the web page 110 in accordance with the specific characteristics of the requesting client device 102A, having knowledge of display 224 capabilities – such as screen size, resolution, and other parameters - useful in determining the way in which the transcoded data is to be displayed on the machine 102A.

[0041] Optionally, the web page 110 or extracted data or both can be stored 312 in database 126. Engine 140 transmits 314 the transcoded data 130 that has been extracted and transcoded to a target format from web page 110, in accordance with the schema 122, to the requesting client machine 102A. As noted above, transcoded data 130 may comprise transcoded navigational data for menu application 82 and informational content data (e.g. a list of products and related information from a web page) for displaying by browser application 86.

[0042] Signature schemas are pre-defined documents, and may be eXtensible Markup Language (XML) documents utilizing an SQL-like query language, to incorporate instructions and data with which to intelligently extract the data from web pages (which web pages are typically coded in HTML, DHTML, XHTML, XML, RSS, JavaScript, etc). This extracted data may be transcoded and provided to client machines 102, or used to dynamically generate a relational database (e.g. 126) or both. Each signature schema incorporates an understanding of a particular web site's data including relationships among the various data (e.g. among its primary informational content found in the body of its web pages as well as among such content and associated navigational data (e.g. web page links) that govern the data in the page). As described further herein below, prior knowledge of the web page code including specific identifiers, tags and text (i.e. strings) used within the code (sometimes referred to as "signatures" herein), may be used to define instructions to identify portions of the code of interest and to extract specific data.

[0043] As a further feature, transcoding may be configured to provide continuity of browsing/transactional/session experience enabling a user to switch client machines (e.g. starting with client machine 102A and switching to machine 102B (or vice-versa)). A user may be enabled to start an interaction with a web site and have displayed data (published

content and navigational data) on the client machine 102A. The browsing session may then be continued on a second client machine (102B) while retaining the transcoding as provided to the first client machine. For example, a user on a desktop can continue to browse the published content and navigational data of the web site as previously experienced on a mobile device, using only a portion of the desktop screen (for example) for data display.

[0044] In accordance with the present embodiment, a signature schema document may be defined for all the pages of a particular web site. Large data-driven web sites (e.g. 104) don't maintain thousands of individual web pages per se. The sites typically adopt a few page family templates 108 and dynamically populate these with pertinent content from database 112 comprising information (e.g. weather, stock data, news, shopping/product data, patent data, trade-mark data etc.) as applicable when a client requests a particular page. Each template represents a family of pages having objects and attributes. Below are representative example page family templates and their objects and attributes for a web site offering news and an e-commerce web site offering products for sale electronically:

Example 1: News site
Family: List Page
Objects: lists a selection of news stories
Attributes: Title, abstract and date
Family: Detail page
Objects: lists a single news story (and optionally other related stories)
Attributes: Journalist, City, Date, Title, Full Story, Image

Example 2: E-commerce site
Family: List Page
Objects: lists a selection of products
Attributes: Image, Item Name, Price, Sale Price
Family: Search Page (a specific kind of list page)
Objects: Similar to a list page

Attributes: Similar to a list page

[0045] Each family of pages (the family template) can be identified by a "signature" or unique set of one or more features that automatically identifies a given page on a web site as part of the family and differentiates that family from another family of pages. Similarly each object and attribute field of interest can be identified with its respective unique signature within a family of pages. A signature schema document typically comprise numerous pieces of information (commands), for example, information that instructs the engine 140 for:

identifying all page families;
identifying and extracting a subset of data (i.e. desired objects and attributes) for each page family;
capturing the (implicit or explicit) relationships between the objects and attributes; and
transcoding the data.

[0046] A signature schema document may also be configured to enable special functionality for the target web site including searching, logging in a user, purchasing items, etc.

[0047] In accordance with a present embodiment, the structure and syntax of a representative signature schema document for a representative e-commerce site eshop.ca is shown and described. Engine 140 may be configured to receive web page code comprising text data and search through the text in accordance with the schema document instructions that provide SQL-query like language instructions. Engine 140 maintains a pointer within the text as it moves through the web page code performing various actions, as described below, in accordance with the schema instructions. Table 1 illustrates a snippet of a representative signature schema:

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <site>
3       <version major="1" minor="2"/>
```

```
4     <url location="http://www.eshop.ca" key="eshop.ca" name="E-Shop" />
5     <advanced>
6
7           <index_link value="http://www.eshop.ca/home.asp" />
8     </advanced>
9     <page_type>
10          <lookup type="pex" action="locate_string" name=
                  "list_elements" id="mylist_1" ref="Compare products"
                  alt1="Sort products" />
11          <lookup type="pex" action="locate_string" name="item_elements"
                   id="myitem_1" ref="&quot;product-details&quot;" />
12          <lookup type="pex" action="locate_string" name="menu_elements"
                  id="mymenu_2" ref="anc-lhsnav-subItem" />
13          <lookup type="pex" action="locate_string" name="menu_elements"
                  id="mymenu_1" ref="product-table" />
14          <lookup type="pex" action="locate_string" name="item_elements"
                  id="myitem_1" ref="*" />
15    </page_type>
16    <list_elements id="mylist_1">
...
17    </list_elements>
...
18    <item_elements id="myitem_1">
19          <actions>
20                <lookup type="pex" action="move_ptr" ref="&lt;/head&gt;" />
21          </actions>
22          <element>
23                <lookup type="pex" action="get_string" name="image"
                        ref="largeimageref" location="after" start="&lt;img src=&quot;"
                        end="&quot;" />
24                <lookup type="pex" action="get_string" name="title" ref="product-
                        details-prd-title" location="after" start="&lt;span"
                        end="&lt;/span&gt;" include_sz="1" strip_tags="1" />
25                <lookup type="pex" action="get_string" name="price" ref="our price:"
                        location="after" start="&lt;td" end="&lt;/td&gt;" include_sz="1"
                        strip_tags="1" />
26                <lookup type="pex" action="get_string" name="sale_price"
                        ref="sale price:" location="after" start="&lt;td" end="&lt;/td&gt;"
                         include_sz="1"  strip_tags="1" tolerance="1" />
27                <lookup type="pex" action="get_string" name="description"
                        ref="detailbox-text" location="middle" start="&lt;p"
                        end="&lt;/p&gt;" include_sz="1" strip_tags="1" />
28          </element>
```

29	</item_elements>
...	

Table 1 - XML Signature Schema Snippet for E-Shop.ca

[0048] In the XML code snippet of Table 1, instructions at line 4 are for verifying that the web page under consideration and the signature schema relate to the same web site/domain – eshop.ca. Instructions at lines 9-15 are for determining the particular page family to which the web page under consideration belongs. A respective signature that defines the particular page family has been previously identified for use to distinguish the page. The engine 140 processes the <page type> tag by registering the identification strings for each page family. When a web page is obtained by the engine as input, the engine may be able to identify the page family by its unique string ref=" and the command provides the related tag within the signature schema document where further instructions for the particular web pages are found:

[0049] **action="locate_string":** command to check for the existence of a string.

name=": identifies the type of page family for each identified family.

id=": assigns an id to the page family that is used across the signature schema document.

[0050] For example, at line 10, the instructions identify a web page using the alternative signatures "Compare products" or "Sort Products". Web pages with these strings are of the same family type. The instructions at line 10 provide a reference tag to further instructions for this family, providing a link to instructions for the list_elements page family with and ID of mylist_1 (see lines 16-17). Similarly the other lookup instructions provide references to the specific instructions within the signature schema document for handling a web page of each web page family. Representative instructions for some of the web page families are provided in Table 1, for example, at lines 16-17 and 18-29 with others omitted for brevity.

[0051] With reference to the extraction instructions for one of the web page families (e.g. item_elements id="myitem_1") at lines 18-29, the instruction at line 20 advances the scan pointer within the text file of the web page code to a beginning limit of a region of interest

indicated by a signature reference. This establishes an upper limit for review within the text file. Though not shown in this table, an end limit may be defined as well (See Table 4). Further such instructions at lines 22-28 may comprise commands to locate the subset of data using "signatures" such as string identifiers that uniquely identify the data within the region of interest. In the present example the instructions locate and extract a plurality of elements, namely, product image, title, price, sale price and description for a product of the item web page family. For example, instructions at line 23 extract a string in between the first "<img src="" and """ that appears after next appearance of "largeimageref". The string returned is the path (relative URL at web site eshop.ca) to the product image. By advancing a search scan pointer within the web code to a particular location, references before that location can be skipped when searching. Any prior instances of a signature string such as "largeimageref" may be ignored. In this way, otherwise ambiguous signature references can be avoided.

[0052]The example in Table 1 shows at least some of the instructions (e.g. lines 23 -27) including one or more directional references relative to the signatures to locate and extract the subset of data. For example, directional references such as "before" or "after" command the engine to extract the data that is in a relative position in the web page before or after the signature string (i.e. ref=). Moreover, such instructions may further include at least one of a start reference or an end reference further pinpointing the location of the data in accordance with that direction. Additional directional reference information is discussed herein with reference to code snippets in other Tables and the discussion of an embodiment of signature transcoding engine syntax presented below.

[0053]The example within Table 1 demonstrates the extraction of data and the establishment of relationships between objects and elements within a same page of a web site. However, signature schema documents may further capture relevant attributes of an object across pages. For example, a user of client machine 102A may click through a number of web pages in eshop.ca to get to a specific product page (e.g. Department -> Product Category -> Product Sub-Category -> Specific Product, such as TV & Video > 19"-

21" TVs > LCD TVs > BrandX Product. The navigational hierarchy representing a categorization may be captured and associated to the extracted objects and there elements.

[0054] For brevity, certain instructions were omitted from Table 1. Tables 2-4 provide representative instructions for further web page families for e-shop.ca that may be read with Table 1. Table 2 below provides representative instructions, e.g. for lines 16 and 17 of Table 1, including instructions for a web page family related to a list of items/products for sale. Whereas instructions at lines 22-28 provided product data extraction instructions for a web page family showing a single item (i.e. product), the instructions of Table 2 provide additional instructions that repeat product data extractions for each product in the list.

```
1   <list_elements id="mylist_1">
2       <paging>
3           <page_variable value="page" />
4           <page_start value="0" />
5           <lookup type="pex" action="get_string" name="link"
                ref="Next&amp;nbsp" location="before" start="&lt;a
                class=" end="&lt;/a&gt;" include_sz="1" strip_tags="1" />
6       </paging>
7       <actions>
8           <lookup type="pex" action="move_ptr" ref="Sort or compare products"
                ref_alt_1="Sort products" />
9       </actions>
10      <element>
11          <lookup type="pex" action="get_string" name="link" ref="thumbnail"
                location="before" start="&lt;ahref=&quot;" end="&quot;&gt;" />
12          <lookup type="pex" action="get_string" name="image" ref="thumbnail"
                location="middle" start="&quot;" end="&quot;" />
13          <lookup type="pex" action="get_string" name="title"
                ref="class=&quot;tx-strong-dgrey&amp;quot;" location="after"
                start="&lt;a href=" end="&lt;/a&gt;" include_sz="1"
                strip_tags="1" />
14          <lookup type="pex" action="get_string" name="price" ref="pricepill/"
                location="after" start="/"  repeat_start="1" end=".gif"
                tolerance="1" />
15          <lookup type="pex" action="move_ptr" ref="pricepill/" />
16      </element>
17  </list_elements>
```

Table 2 - XML Signature Schema Snippet for Product List Page Family of E-Shop.ca

[0055] If the engine 140 identifies that the page is of the "mylist_1" family, the engine determines the location in the signature schema document that contains the signature for the objects and elements of that family and applies the instructions therefor. A product list at e-shop.ca may span multiple web pages. Instructions at lines 2-6 of Table 2 find the number of pages and generate the links for each of the pages. Instructions at lines 7-9 (action tag) advance the search scan pointer to the region of web page code that may be of interest (i.e. in this case, the start of the list). In this way, a local signature reference can be used and any earlier ambiguous references skipped. Skipping to the local region of interest may also make the specification of the signature reference less complicated.

[0056] Taking advantage of inherent repeated patterns in the web page code, instructions at lines 10-16 (elements tag) of Table 2 provide product data extraction instructions that may be repeated for each product in the list. The engine 140 may be provided with commands to scan for each data element of interest using a signature reference e.g. ref=", an action, one or more positional instruction(s) to further identify the data within the text of the web page code, and any additional text data manipulation instructions to extract the data (e.g. to remove HTML formatting characters or add characters). The instruction at line 15 moves the scan pointer to the end of the object (in this example a product in a list of products) to ready the instructions for application against the next object (product) in the list.

[0057] More particularly:

lookup type="pex": string lookup
action ="get_string": returns a value back that is the desired element of the object.
name="link": the object element, in this case the link to the product page
ref="thumbnail": the reference string that identifies where to find the value of the link
location="before": the value of the link is before the ref string
start="<a href="": look for the ref string after this value
end="">": look for the ref string before this value.

```
1 <search_elements id="mysearch_1">
2     <settings>
3         <search_path value="http://www.eshop.ca/search/search.asp/>
4         <search_variable value="keyword" />
5     </settings>
6     <paging>
7         <page_variable value="page" />
8         <page_start value="0" />
9         <lookup type="pex" action="get_string" name="link" ref="Next&amp;nbsp"
              location="before" start="&lt;a href=" repeat_start="1"
              end="&lt;/a&gt;" include_sz="1" strip_tags="1" />
10    </paging>
11    <actions>
12        <lookup type="pex" action="move_ptr" ref="bg-compare-hero" />
13    </actions>
14    <element>
15        <lookup type="pex" action="get_string" name="link" ref="&gt;"
              location="after" start="&lt;a href=&quot;" end="&quot;&gt;" />
16        <lookup type="pex" action="get_string" name="image" ref="&lt;a href"
              location="after" start="&lt;img src=&quot;" end="&quot;" />
17        <lookup type="pex" action="get_string" name="title"
              ref="class=&quot;tx-strong-dgrey&amp;quot;" location="after"
              start="&lt;a href=" end="&lt;/a&gt;" include_sz="1" strip_tags="1" />
18        <lookup type="pex" action="move_ptr" ref="bg-compare-hero" />
19    </element>
20 </search_elements>
```

Table 3 - E-Shop Search Family Signature Schema Snippet

[0058] If the engine 140 has identified that the page is of the "mysearch_1" family the engine applies the portion of the signature schema document that contains the signature for the objects and elements of that family, shown above in Table 3.

<settings>...</settings>: Contains any web page specific manual overrides such as excluding certain menu items, customization, modification of a menu that may be desired. In this example, as per line 3 a value of form variable "keyword" will be posted to "http://www.eshop.ca/search/search.asp".

<paging>...</paging>: Manages paging for the search pages.

<actions>...</actions>: Instruct the engine to move the scan pointer to the string "bg-

compare-hero" (line 12 of Table 3) and start looking for elements from there.

<element>...</element>: Contains lookup instructions for each object element as previously described.

```
1 <menu_elements id="mymenu_1">
2       <settings>
3               <black_list value="Site Index##External Link" />
4       </settings>
5       <actions>
6               <lookup type="pex" action="move_ptr" ref="bg-lhsnav-title" />
7               <lookup type="pex" action="end_ptr" ref="&lt;/table&gt;" />
8       </actions>
9       <element>
10              <lookup type="pex" action="get_string" name="link" ref="&lt;li&gt;"
                        location="after" start="&lt;a href=&quot;" end="&quot;" />
11              <lookup type="pex" action="get_string" name="title" ref="&lt;li&gt;"
                        location="after" start="&lt;a href=&quot;" end="&lt;/a&gt;"
                        include_sz="1" strip_tags="1" />
12              <lookup type="pex" action="move_ptr" ref="&lt;/li&gt;"/>
13      </element>
14 </menu_elements>
```

Table 4 - E-shop Menu Family Signature Schema Snippet

[0059] If the engine 140 has identified that it is looking for a menu on a page that contains the menu style of the "mymenu_1" family, the engine applies the portion of the signature schema document that contains the signature for the objects and elements of that family, shown above in Table 4.

<settings>...</settings>: Contains any page specific manual overrides such as exclude list, customization, modification, personalization, etc. In this example, as per line 3, any result that matches "Site Index", "External Link" are excluded but partial matches are also possible by using wild card strings.

<action>...</action>: Lines 6 - 7 of Table 4 sets the start and end limits to instruct the engine 140 where to look for menu items.

<element>...</element>: Contains lookup instructions for each object element as

previously described. In this example, lines 10 and 11 of Table 4, an element in 'mymenu_1' (each individual menu entry of web page) contains link and title as its properties. Line 12 instructs the engine to move the pointer to "
" to get ready to loop through and extract the next menu item with the same elements, taking advantage of the repeated patterns within the text of the web page code.

[0060] Though the example described relates to extracting informational content for an e-commerce oriented site, no limitation should be applied. Similar instructions may be defined for other types of sites, for pages which permit a user to input information and for navigational data extraction.

[0061] Signature schema document 122 may further comprise transcoding instructions (not shown) for use by engine 140 to express the extracted subset of data in a target format (e.g. a format of HTML, XML, script etc.) for use by the requesting client machine 102. For example, the transcoding instructions may define a web page for displaying the extracted data in browser application 86 that is suitable for display on the client device 102. The formatting rules can be system and/or user defined and can include parameters such as but not limited to: object positioning, object colour, object size, object shape, object font/image characteristics, background style, and navigational item display (e.g. in a menu as described above) or for display with the content in the generated page on the client screen. Browser application 86 (e.g. of machine 102A) may be configured for using a markup language (e.g. cHTML) or other code format that is not identical to the code provided by web page 110. Alternatively, transcoding instructions may be defined to express the extracted subset of data in XML or another code format such as for use by a different client application or plug-in to a client application such as menu application 82 or another application (not shown) on client machine 102.

[0062] Signature schema documents may be prepared (i.e. coded) using a computing device such as computing device 128. Computing device 128 may be any suitable desktop or laptop device capable of coding documents (which may be but need not be XML-type documents) and may be configured to automate or semi-automate coding of

such documents.

Computing device 128 may be coupled to web site 104 to retrieve web pages from the site for reviewing to prepare the custom signature schema document for the site. Computing device 128 may be configured to automatically review the web page code and apply heuristics or other techniques (e.g. spatial analysis) to determine probable content of interest (i.e. subset of data) and generate code to extract the subset of data. For example, primary content of interest tends to be located toward the centre of the web page. In another embodiment, the computing device may facilitate a user coding signature schema to manually assist with the analysis of the web page and identification of subset of data and the generation of the instructions. Computing device 128 may be further coupled to repository 124 to provide (e.g. up-load or publish) coded signature schema documents for use by server 120.

Automatic Generation of Signature Schema 122

[0063] Referring to Figures 1 and 7, in one embodiment, the computing device 128 of system 100 comprises an automatic signature generation tool 150 for preparing a custom signature schema document for web pages of a web site. Computing device 128 may further comprise a desktop tagging tool 160 having a graphical user interface 165 (which may be adapted to cooperate with a web browser application) for assisting a user to identify the desired data (e.g. product title, image, description and price data) in a web browser window 700. User interface 165 may comprise a portion of the window while the remaining portion 702 displays the rendered web page 110A for which a signature schema is to be constructed. User interface 165 may present a form 706 showing the desired data (fields and values therefor) where candidate data values 710 from data 704 of the web page 110A populates the form 706. User interface 165 may facilitate confirming or amend the candidate data values. For example, data replacing the candidate data 710 may be selected and captured (not shown) from the rendered web page 110A through "drag and drop" or highlighting/copying user gestures.

[0064] User interface 165 may be predefined to present candidate desired data (i.e. for

particular desired data types that are expected to be found on web pages for such web sites of a similar genre). That is, a user interface 165 for an e-commerce web site selling products may be defined to present "product title", "image", "price", etc. If a particular candidate value was incorrect, for example, product image 704A, title 704B, etc., such may be selected and dropped or copied into form 706 of interface 165. Optionally, the interface may permit the user to add data types (fields and field values) to the presented data. In association with these actions, tool 160 examines the associated HTML source code/tags of the rendered web page for capturing this data. Desktop tagging may be useful to assist with the identification of the desired data within the web page code so that signatures therefor within the web page code of similar pages may be determined for defining the signature schema documents.

[0065] Although the desktop tagging tool 160 and the automatic signature generation tool 150 are described in relation to computing device 128, it will be understood that any one of the client machines 102 may be configured to comprise the tools 150 and 160. Further, it will be understood that the exemplary operation of the automatic signature tool 150 may be implemented similarly on the client machines 102. Similarly, the flow of interactions may apply similarly for either one or both of the computing device 128 or the client machine 102.

[0066] Although signature schema documents 122 may be manually coded, these activities may be time consuming and subject to human error. Therefore, by providing an automatic signature tool 150 to automate coding of signature schema, transcoded web pages (e.g. 130, 132), and thus transcoded web sites, may be readied for use faster and more reliably.

[0067] Referring to Figure 6 shown are exemplary operations 600 of the automatic signature generation tool 150. A detailed example of two sample web pages used to define a signature schema will be presented below. At 602, the automatic signature generation tool 150 receives an input identifying the desired data that is to be located and extracted, that is, for which signatures and instructions are desired. Tool 150 receives an input identifying a set of fields and corresponding field values for extraction from at least two

sample web pages of a web page family. The fields and field values have also been referred to as elements herein. That is, the fields may refer to the categories or attributes by which an object (such as an item for sale) may be defined. For example a product object such as a camera may have the following fields: image, title, price, description. The values for each of the fields related to the camera may be referred to as field values. The field value for the title field may include "BrandX 7.2MP Digital Camera".

[0068] The input identifying the fields and field values for extraction as defined in 602, may be provided by: a manual review of the web page to identify desired fields (e.g. locating the desired image within object tags of a web page) and to indicate the content type of various tags in the web page (e.g. navigation, title, price, image, item description, etc.). Alternatively, the input fields and field values of step 602 may be semi-automated using the desktop tagging tool 160 to highlight portions on the web page and therefore visually select which content data corresponds to what meaning (e.g. to select the elements on a page linked to a field). Further alternately, the desktop tagging tool 160 may be used to automatically populate fields and estimated values for the fields and to allow a user to confirm / correct estimated fields (e.g. by using heuristics or other rules automatically applied in combination with pre-defined locations of fields (e.g. confidence intervals) to web pages to identify likely data) provided by tagging tool 160 or other module (not shown).

[0069] At 604, each identified field and corresponding field value is located within object tags of each of the at least two sample web pages. For example, if for the first sample web page, the input received identifies an image field having the value "product_image.gif", then this value is first located within an object tag of the first web page. For example, the object tag may be: <img src="product_image.gif"> and it is the second image object tag (e.g. a second instance of the <img tag) in the code of the web page that contains this object tag. The received input further identifies that for the second web page, the identical field type (e.g. image field) has a value of "sample_image.gif". It is determined that the second image object tag in the code of the web page is associated with the identified image field (e.g. the object tag may be <img src="sample_image.gif">).

[0070] At 606, the automatic signature generation tool 150 compares the object tags of identical fields (e.g. image field) between the two sample web pages to identify a commonality between the object tags for the identical fields (such as common location, string identifiers, attribute type, and other patterns (i.e. a pattern comprising characters that describe a set of strings that can uniquely identify a field value)) within the plain text (ASCII) web page code. A pattern may include "string1"*"sting2" where * represent 0 or more characters between the characters of "string1" and "string 2".

[0071] In the above example, the commonality between the two identified object tags may be that the object tag was the second instance of the "img" attribute within the code of each web page; that the object for the two sample web pages starts with `src="` and that `"` ends the string that provides the field value for each object. For example, the object tag of the first web page provided the string "product_image.gif". Further, the object tags corresponding to each web page and having the identical image field type may be identified by the attribute "<img".

[0072] At 608, automatic signature generation tool 150 uses the commonality between object tags of identical fields of the two sample web pages to define instructions to locate and extract the desired data, which instructions comprise a portion of the signature schema 122 for web pages of the same family type. Operations 600 may be repeated for each of the identified fields and field values (elements) to determine the commonality and patterns between the two sample web pages, in turn defining signatures and instructions with which to define at least a portion of signature schema 122. Further, operations 600 may be repeated for other web pages of other family types in the web site to generate the instructions to code other respective portions of schema 122.

[0073] An example of the operations 602, 604, 606, and 608 will now be described with reference to two illustrative sample web pages (and their illustrative HTML code in Table 6). As described earlier, pre-identified fields and field values indicating the subset of data to be located and extracted from the web page code for this web page family are provided (Table 5) for each of the sample web pages (for operations 602). As also described, the pre-defined fields may either be identified manually by the user or using the desktop tagging

tool 160 including estimated locations of the fields to generate the desired fields and field values.

Item1	
Image	Product_image.gif
Title	Product Title
Price	$79.99
List Price	$99.99
Description	This is a description for Product title made by Product Manufacturer
Item2	
Image	Sample_image.gif
Title	Sample Title
Price	$99.99
List Price	$109.33
Description	This is a description for Sample title made by Sample Manufacturer

Table 5 - Example Fields and Field Values of Two Sample Web Pages

Item1.html

```
<html>
<head></head>
<body>
<img src="company_logo.gif" class="image" />
<div class="product">
<h1>Product title</h1>
<h2>Product Manufacturer</h2>
<img src="product_image.gif" />
<br>
List Price: <strong> $99.99 </strong>
<br />
<br>
Our  Price: <strong> $79.99 </strong>
<br />
<p>
This is a description for Product title made by Product Manufacturer
</p>
</div>
</body>
</html>
```

Item2.html

```
<html>
<head></head>
<body>
<img src="company_logo.gif" class="image" />
<p>
disclaimer
</p>
<div class="product">
<h1>Sample title</h1>
<h2>Sample Manufacturer</h2>
<img src="sample_image.gif" />
<br>
List Price: <strong> $109.33 </strong>
<br />
<br>
Our  Price: <strong> $99.99 </strong>
<br />
<p>
This is a description for Sample title made by Sample Manufacturer
</p>
</div>
</body>
</html>
```

Table 6 - Example HTML Web Pages Document of the Two Sample Web

[0074] As noted, automatic signature generation tool 150 repeats operations 602-608 for each of the input fields (e.g. image, price, title, description) to define a commonality between the web page code (e.g. tags etc.) used to describe each of the respective fields and thereby define the signature schema 122 for that field.

Step 1 – Identify the Image Field and Field Value in the Sample Web Pages

[0075] First, the automatic signature generation tool 150 examines the web page code of Item1 for the identified image field . Tool 150 may initially identify "src" as an attribute corresponding to the image field and scan the source (HTML document) of the Item1 web page for src="product_image.gif". It does find a match (as it ought to since the field was previously selected from this code) and the location thereof. It then scans item2 but no match is found in item2. Next the automatic signature

generation tool looks at "<img " within the source document of item1. It determines that it is the second match (corresponding to the "img" attribute). Once the object tag corresponding to the image field and field value has been located within item1, the automatic signature generation tool 150 examines Item2 for the image field having value of sample_image.gif. When looking at Item2, the second object tag having the image attribute also provides the object that contains the image. Now that a matching object corresponding to the same field type (e.g. image) has been found, a similar heuristic is used by the automatic signature generation tool 150 to locate the result from within the object. If the object is a text node, the process is complete. Text nodes are easily distinguished by their respective tags in the code. Otherwise, the start and end of the object corresponding to the image field may need to be located. Using pattern recognition techniques, it is found that the pattern ` src="` starts the string and that `"` ends the string. Therefore the following entry may be added to the signature schema 122 for defining the image field.

<lookup type="pex" action="get_string" name="image" ref="<img " repeat_ref="1" start="
src="" end=""" />

Step 2 – Identify the Title Field and Field Value for each Sample Web Page

[0076] From Item1 the object <h1>Product title</h1> is selected by the automatic signature generation tool 150 based on the identified fields to review. Tool 150 identifies that it is a text node within the code and looks to its parent to identify uniqueness. There are no attributes for the parent <h1>. Next the automatic signature generation tool 150 looks at "<h1" within Item1. It determines that it is the only match. When looking at Item2, there is only one match, and the matching object tag contains the title. Now that the automatic signature generation tool 150 has obtained the matching object for the title field in each of the sample web pages, a similar heuristic is applied to locate the result from within the object. Since the object is a text node, the process is complete. Therefore the following entry may be added to the signature schema 122 for defining the title field of a web page.

<lookup type="pex" action="get_string" name="title" ref="<h1" start=">" end="<" />

Step 3 – Identify the Price Field and Field Value for each Sample Web Page

[0077] From Item1 the object $79.99 is selected by the automatic signature generation tool 150. There are no attributes to be checked for this element. Next the element looks at "<strong" within Item1. It determines that it is the second match that contains the desired price ($79.99). When looking at Item2, the second strong tag also provides the object that contains the price. Since the object is a text node, the process is complete. Therefore the following entry may be added to the signature schema 122 for defining the Price field of a web page:

<lookup type="pex" action="get_string" name="price" ref="<strong" repeat_ref="1" start=">" end="<" />

Step 4 – Identify the List Price and the List Price Value for each Sample Web Page

[0078] From Item1, the object $99.99 is selected by the automatic signature generation tool 150. There are no attributes to be checked for this element. Next the signature generation tool 150 looks at "<strong" within Item1. It determines that it is the first match that corresponds to the selected object that contains the desired list price field and value. When looking at Item2, the first strong tag also provides the object that contains the list price field and value. Since the object is a text node, the process is complete. Therefore the following entry would be added to the signature schema 122 for defining the List Price field of a web page:

<lookup type="pex" action="get_string" name="price" ref="<strong" start=">" end="<" />

Step 5 – Identify the Description and the Text Value for the Description field for each Sample Web Page

[0079] From Item1 the next identified field for automatic signature generation tool 150 is object " provides a description for Sample title made by Sample Manufacturer ". This object represents the pre-identified Description field and field value of Item1. There are no attributes to be checked for this object. Next the signature generation tool 150

looks at "<p" within Item1. It determines that it is the first match that contains the desired description field and field value. When looking at Item2, the first <p tag does not provide the object that contains the desired description (e.g. "This is a description for Sample title made by Sample Manufacturer"). The parent object is selected next by the automatic signature generation tool. It identifies the attribute class="product", and scans Item1, and determines that it is the only match. The <p tag is processed again, limiting its search to the parent. The <p tag is identified as the first instance within the parent in Item1. Next the same process is performed on Item2. First the attribute class="product" is located. The first <p tag that is a child of the object containing class="product" is found. The <p object also contains the desired description (e.g. "This is a description for Sample title made by Sample Manufacturer". Since the object is a text node, the process is complete. Therefore the following entry would be added to the signature schema 122 for defining the description of a web page:

<lookup type="pex" action="get_string" name="description"
ref="class="product"" start="<p>" end="<" />

[0080] Accordingly, as illustrated in Step 5 of the example above, in one embodiment, the automatic signature generation tool 150 examines the HTML document (or other format of web page) and constructs a programmatic data structure to model a hierarchy of the tags. The resulting structure may be a tree, which defines the parent, siblings and children of each object. The operations may identify the key objects that contain the data required for the signature schema document 122. Once a particular object is identified as being a desired data field (i.e. is one of the fields in Table 5), the uniqueness of the object may be identified by examining its properties (for example class, style, id) within the structure. If the properties of the object are not unique, then the task to identify the uniqueness for the object would expand to its parent, siblings and children. For example, if the object is a text node of the tree (or other hierarchical structure), the object may use the properties of its parent to assist with the identification of is uniqueness for expression as a signature. The operations may expand in all directions uniformly (i.e. examine parent, then previous sibling, then next sibling, then first child). The properties of each of these items may also

be merged with the desired object to build out the uniqueness. This process would then be repeated on the parent, then the previous sibling, etc, until a unique identifier was found. Once a unique identifier was found, an expression would be created for the signature.

[0081] Accordingly, in view of the above, the automatic signature generation tool 150 provided by the computing device 128 provides the signature schema 122 for a new web page family using at least two sample web pages. As illustrated in steps 604 and 606, the tool 150 compares two or more delimiters (pertaining to a common schema of the definition of the pages) from each of the sample web pages in order to identify common uses of the delimiters (and their contents). Once identified as a match, the corresponding object, for example, is placed in the hierarchical structure (or other ordered list, etc.) for defining the signature schema 122.

[0082] It is recognized that the hierarchy can link entities either directly or indirectly, and either vertically or horizontally. The only direct links in a hierarchy, insofar as they are hierarchical, can be to the entities' immediate superior or to the entities' subordinates, although a system that is largely hierarchical can also incorporate other organizational patterns. Indirect hierarchical links can extend "vertically" upwards or downwards via multiple links in the same direction. Traveling up the hierarchy to find a common direct or indirect superior, and then down again can nevertheless "horizontally" link all parts of the hierarchy, which are not vertically linked to one another. Further, the structure may also be a list implemented using arrays or linked/indexed lists of some sort. The structure may have certain properties associated with arrays and linked lists.

[0083] Further, it is recognized that the structure would be represented in the signature file 122 as the entries or instructions as noted above. It is recognized that a user of the device 128 could manually amend or otherwise review the automatically generated signature file 122, as desired.

[0084] It will be apparent to a person of ordinary skill in the art that as a web site may be re-designed or otherwise changed such that the code of one or more web page families may be changed or a family added, an existing signature schema may require re-coding to

account for the change/addition, as applicable.

[0085] It will be apparent to a person of ordinary skill in the art that as a web site may be re-designed or otherwise changed such that the code of one or more web page families may be changed or a family added, an existing signature schema may require re-coding to account for the change/addition, as applicable.

Signature (Transcoding) Engine Syntax

[0086] In accordance with a present embodiment, further details concerning the syntax of schema instructions are described.

Lookup Syntax

[0087] The lookup tag instructs the engine 140 to perform an insert, delete or query the document contents.

[0088] **Type:** Defines the data type of the lookup. Type may be "pex" for a string expression. Type may also support more advanced options such as regular expressions, API calls, and SQL queries.

Action:

Action = "locate_string": Look for a string ("ref" identifier") value within the data. Return true iff the string exists in the data (i.e. the "ref" identifier index >= 0).

Action = "replace_string": Replace a string within the data with the "ref" identifier.

Action = "move_ptr": Remove all characters in the data that exist before the location of the "ref" identifier.

Action = "end_ptr": Remove all characters in the data that exist after the location of the "ref" identifier.

Action = "get_string" Extract a string based on the location of the "ref", "start", and "end" identifiers.

ID: ID is an identifier of another section within the signature. It allows the result of a query to trigger another set of actions within the signature. This is primarily used when identifying page types. Once a match has been made, specific instructions are executed that are marked with this ID. Recursive data structures (e.g. lists within lists) may also be supported.

Ref: Ref defines the initial identifier that the lookup searches for. If an AND case is required multiple ref identifiers can be used (i.e. ref="string1" ref1="string2"). If an OR case is required ref_[ref identifier] _alt_1 can be used (i.e. ref="string1" ref_alt_1="string2"). To demonstrate (X="1" || Y="2") && (A="8" || B="9") would translate to ref="1" ref_alt_1="2" ref1="8" ref1_alt_1="9".

Repeat_[identifier]: Repeat executes the identifier query additional times. For example, if ref="hello" to set the identifier index at the second occurrence of hello the following tag would be added: repeat_ref="1".

Location:

Location = "before": Search the data in a reverse direction, starting from the "ref" identifier. This implies that both the "start" and "end" identifier indexes must be less than the "ref" index.

Location = "middle": Search the data in two directions, starting from the "ref" identifier. This implies that the "ref" identifier index is greater than the "start" identifier index and less than the "end" identifier index.

Location = "after": Search the data in a forward direction, starting from the "ref" identifier. This implies that both the "start" and "end" identifier indexes must be greater than the "ref" index.

Start: Start is primarily used when action="get_string" and may also be used for replace/remove instructions. The start identifier index will be the start index of the string to extract. If an AND case is required multiple "start" identifiers can be used (i.e. start="string1" start1="string2"). If an OR case is required start_[start identifier] _alt_1 can be used (i.e. start="string1" start_alt_1="string2"). To demonstrate (X="1" || Y="2") &&

(A="8" || B="9") would translate to start="1" start_alt_1="2" start1="8" start1_alt_1="9". To find the n^{th} match see the repeat syntax.

End: End is primarily used when action="get_string" and may also be used for replace/remove instructions. The end identifier index will be the end index of the string to extract. If an AND case is required multiple "end" identifiers can be used (i.e. end="string1" end1="string2"). If an OR case is required end_[end identifier] _alt_1 can be used (i.e. end="string1" end_alt_1="string2"). To demonstrate (X="1" || Y="2") && (A="8" || B="9") would translate to end="1" end_alt_1="2" end1="8" end1_alt_1="9". To find the n^{th} match see the repeat syntax

Max_Index: Max_Index is used to limit the scope of a query by ensuring that no other identifier index is greater than the "max_index". . If an AND case is required multiple "max_index" identifiers can be used (i.e. max_index="string1" max_index1="string2"). If an OR case is required max_index_[max_index identifier] _alt_1 can be used (i.e. max_index="string1" max_index_alt_1="string2"). To demonstrate (X="1" || Y="2") && (A="8" || B="9") would translate to max_index="1" max_index alt_1="2" max_index ="8" max_index _alt_1="9". To find the n^{th} match see the repeat syntax.

Max_Index_Use_Ref: Max_Index_Use_Ref is a Boolean value set to 0 or 1. It is used with Max_Index. When set to 0, the "max_index" will begin querying at the beginning of the data. When set to 1, the "max_index" will begin querying from the "ref" identifier index.

Gbl_append_[identifier]: Gbl_append appends a string passed via the url to the identifiers query value

Gbl_Repeat_[identifier]: Gbl_Repeat executes the identifier query additional times. For example, if ref="hello" to set the identifier index at the second occurrence of hello the following tag would be added: gbl_repeat_ref="var" where var would be passed in the URL i.e. http://www.eshop.ca/mobile/fatfree.asp?site=...&url=...&var=1.

Tolerance: Tolerance is a Boolean value set to 0 or 1. It is used to return an empty string. By default tolerance is set to 0 which enforces that a property be found on a page, otherwise the page will be marked as "invalid" and an appropriate error message returned.

When set to one, an empty value is returned for properties that can not be located.

Include_sz: Include_sz is a Boolean value set to 0 or 1 and used with get_string. It is by default set to 0. When set to 1 it includes the "start" value and the "end" value as part of the result.

Include_start: Include_start is a Boolean value set to 0 or 1 and used with get_string. It is by default set to 0. When set to 1 it includes the "start" value as part of the result.

Include_end: Include_end is a Boolean value set to 0 or 1 and used with get_string. It is by default set to 0. When set to 1 it includes the "end" value as part of the result.

Closetag: Closetag is a Boolean value set to 0 or 1 and used when action="get_string". It appends /> to the extracted value.

Strip_Tags: Strip_Tags removes HTML tags from the value and used when action="get_string".

Strip_tags="1": remove all tags.

Strip_tags="2": remove all br and script tags.

Strip_tags="3": remove all tags except replace </p> with
.

Strip_tags="4": remove all tags except replace
 with
.

Strip_tags="tag1,tag2,...tagN": remove all tag1, tag2,... tagN leaving any tag not listed.

Notrim: Notrim is a Boolean value set to 0 or 1 and used when action="get_string". By default all value have white spaced trimmed. When this property is set to 1, white space is not trimmed.

Append: Append is a string value and used when action="get_string". It appends a string to the extracted value.

Prepend: Prepend is a string value and used when action="get_string". It prepends a string to the extracted value.

Upper: Upper is a Boolean value set to 0 or 1 and used when action="get_string". It converts all characters to upper case.

Lower: Lower is a Boolean value set to 0 or 1 and used when action="get_string". It converts all characters to lower case.

Page Syntax

[0089] The page syntax extracts the paging information from the data. This allows the end user the ability to change pages just as on the desktop.

Page_variable: Defines unique key that defines a family's paging feature.

Page_start: Defines value of first page in a family's paging feature.

Page_post: Path where paging variable(s) must be transmitted to.

Page_start : Defines value of first page in a family's paging feature.

Page_increment: Defines value that paging increases by for each page in a family's paging feature.

Page_block: Defines unique key that defines a family's paging block feature.

Page_block_size: Defines the size of the family's page block. (i.e. 10 items per page)

Url_append: Append the unique key that defines a family's paging feature and the page number.

Search Syntax

[0090] Make a web site family's search feature functional by specifying details such as what variable to post.

Search_path: Search path where search variable must be transmitted to

Search_variable: Name of search variable which a web site's search feature is looking to read, request, post, etc.

Url_replace: Remove a portion of the url that is specific to posting search parameters

URL Syntax

[0091] The url tag defines global properties for a site, including the url, and name:

<url location="http://www.eshop.ca" key="eshop.ca" name="E-Shop" />

Name: Name is the name to display when browsing using the gateway 120

Location: Location defines the fully qualified address of the site.

Key: Key is the site.

Advanced Syntax

[0092] The advanced tag defines global properties for the site. This at a minimum includes the path to the initial page of the site.

<advanced>

<index_link value="http://www.eshop.ca" />

<check_out value="1" />

</advanced>

Index_link: Index_link specifies the path to the initial page of the site. This is usually the same page as the location property from the URL syntax. This field is always required.

Append_link: Appends a string value to every URL requested for this site.

No_purchase: No_purchase is a Boolean value 0 or 1. The default value is 0 which implies that an item should contain a purchase link. When true, the purchase link is removed.

No_item: No_item is a Boolean value 0 or 1. The default value is 0 which implies that Item pages should show up in the breadcrumb. When true, the item is not added to the breadcrumb.

Check_out: Check_out is a Boolean value 0 or 1. The default value is 0 which implies that Item purchase link sends the request and control away from the gateway server 120. When true, then a checkout process has been created for use with gateway server 120.

Product_img_width: Product_img_width defines the width of all item images.

Use_cookies: Use_cookies a Boolean value 0 or 1. By default it is set to 0, and cookies

are not passed to the site. When true, gateway 120 passes all cookies from client machine 102 to the site 104, and from the site 104 to the client machine.

Page Type Syntax

The page type is a collection of lookup queries that have an id associated with them. Lookup queries may be processed in a top down fashion. The first successful lookup will trigger another section in the signature schema document. For example, if the following evaluates to true:

<page_type>

<lookup type="pex" action="locate_string" name="list_elements" id="mylist_1" ref="<!--" />

</page_type>

[0093]Then the tag element <list_elements id="mylist_1"> would be executed next.

General Element Syntax

Elements include list_elements, menu_elements, item_elements, search_elements, form_elements. Each element has an ID. For example a menu element:

<menu_element id="menu_id"/>

The element may contain the following sub containers (settings, actions, elements, paging) which scope resides only within the element. Each element is associated with a specific rendering function.

<menu_element id="menu_id"/>

<settings> </settings>

<paging> </ paging >

<elements> </ elements >

<actions> </ actions >

</menu_element>

Settings Syntax

Settings syntax varies based on the type of element it resides in. Settings allow customizations that only apply to a specific page family.

Black_list – menu_elements: Black_list removes menu items with names that reside in the black list. Each entry is separated delimited (e.g. using two pound characters (##).

Pass_image – list_elements, search_elements: Pass_image adds the image path to the url when requesting an item. The image added to the url will be used as the item image.

Price[n] – item_elements: Price[n] where n is an integer renames the rendered item with name price[n].

Action – form_elements: Overrides the action of a form displayed to the end user.

Handle – form_elements

Handle = "display" - display the form to the end user.

Handle = "post" – post the form.

Handle = "get" – get the form.

Cookie – form_elements: Send additional cookies when posting this form.

Input_[identifier] – form_elements: Input tag adds/modifies a form value with name [identifier] setting its value.

Rename_[identifier] – form_elements: Rename tag renames a form value with name [identifier].

Actions Syntax

The actions tag primary function is data manipulation. It contains lookup queries that modify data with actions of "move_ptr" or "end_ptr".

<actions>

<lookup type="pex" action="move_ptr" ref="</head>" />

</actions>

[0094] Persons of ordinary skill in the art will appreciate that alternative embodiments are contemplated. Though not shown, a client machine may incorporate a transcoding engine, applying a signature schema document obtained from a repository such as repository 124 to web pages received from a web site. For example, client machine 102B may be configured with an engine in cooperation with a mini-browser application or plug-in to another application. The engine obtains the schema document to apply against web page content from a particular web site. Communications with the web site may be direct and not via a gateway 120. The transcoding engine may apply the commands from the schema and transcode appropriately for rendering content by the mini-browser or via the plug-in.

[0095] Figure 4 illustrates a further embodiment comprising a system 400 for content navigation, similar to system 100 of Figure 1 but in which a client machine 102C incorporates a secure transcoding engine 402, for example, for communicating directly with web site 104 via secure communications (e.g. Secure Sockets Layer (SSL) or Transport Layer Security (TLS), etc.). Client machine 102C may be a wireless device such as device 102A or wired device 102B comprising components as described with reference to Figure 2 and as further described with reference to Figure 4.

[0096] Large public database-driven web sites do not typically encrypt data that is publicly available. Instead, the sites encrypt specific pages that contain user information, for example login, signup, checkout, and account management pages. One reason why all content is not encrypted may be that SSL/TLS is resource intensive and reduces scalability. Another reason why all content is not encrypted may be that SSL/TLS increase response times for the end user due to the time spent encrypting and decrypting content. Examples of web sites that follow this model include online stores, news sites, sports information and weather. Therefore, since the number of SSL/TLS pages is relatively small, signature schema can be created to define a mobile friendly layout. Another benefit of the signature schema, is that each field in an HTML form can be classified and populated with user data from an external application. It will be understood that each individual SSL/TLS page will likely require its own respective page family template within a schema.

[0097] In contrast to Figure 1, Figure 4 shows a client machine 102C comprising a browser application 86C similar to browser 86 for communicating with web site 104 via gateway and schema server 120. In a similar way, a signature schema may be used to transcode un-encrypted communications of web pages 110 to provide transcoded data 408. However, browser 86C may be further configured to communicate through secure transcoding engine 402, handing off communications for secure web pages 404 when such communications between machine 102C and web site 104 are to be encrypted. Secure transcoding engine 402 may communicate with gateway and schema server 120 to obtain the signature schema document 122 which may be applied to transcode secure communications with web site 104.

[0098] Figure 5 illustrates a flow among client machine 102C, gateway and schema server 120 and web site 104 for secure communications such as for web page 404. It may be presumed that client machine 102C has previously initiated a flow similar to Figure 3 for a web page 110 that has resulted in transcoded response 408 from gateway and schema server 120 including the actual location of the secure content (e.g. for end to end encrypted communications with site 104 via HTTPs protocols). Browser 86C hands off the request communication (502) to secure transcoder engine 402. Secure engine 402 requests (504) a signature schema 122 from server 120/engine 140. The request may be validated and the schema 122 returned (506) by the engine 140 from schema repository 124 as may be necessary. Secure engine 402 requests 508 the secure content (e.g. 404) via end-to-end encrypted communication from the web server 106. The secure engine 402 receives (510) the secure content 404 from the web server 106, decrypts the content and then invokes the transcoder using the signature schema 122 as instructions to extract the subset of data from the web page 404 and to re-construct the content in a mobile friendly view for rendering by the browser.

[0099] Schema document 122 may include instructions for populating secure responses to web site 104 with data previously stored to client machine 102C. Such information may include personal information that has been stored using an external client application 406

such as a password keeping application for securely storing (encrypted) personal information. Schema documents may be coded with suitable instructions to invoke communications or application programming interfaces between the secure transcoding engine and external application 406 to securely obtain such data. Such information may be available via a plug-in (not shown) to browser 86C.

[0100] System 100 may be implemented so that a plurality of web sites are coupled to the telecommunication network (either alone by a server 106 or by a plurality of web servers like web-server 106), and that a corresponding plurality of schemas for each of those web sites (or each of the web pages therein, or both) can be maintained by gateway and schema server 120 and repository 124. There can in fact be a plurality of gateway and schema servers (like server 120). Client machines 102 can be configured for proxied connection through different servers 120. Servers 120 can be hosted by a variety of different parties, including, for example but without limitation: a) a manufacturer of client machine 102, b) a service provider that provides access to the telecommunication network on behalf of user U of a client machine 102; c) the entity that hosts web-site 104 or d) a third party intermediary. In web site host example it can even be desired to simply combine the web server 106 and schema server engine 120 on a single server to thereby obviate the need for separate servers.

[0101] Accordingly, signature schemas may be defined to provide custom browsing experiences for small (e.g. mobile) devices (among others) and the proposed framework avoids changing web site code for existing web sites. Data extracted from the web sites may be intelligently stored to a relational database using knowledge of the web pages (i.e. the objects and their attributes) incorporated into the signature schemas. Query language may be used to direct a search of the web page as an ASCII text file to look for signatures to distinguish the web page's family (from other web page families of a site) and to identify the subset of data to be extracted.

CLAIMS

1. A method of automatically generating transcoding instructions to locate and extract a subset of data from a selected web page of a web site, the method comprising:

receiving an input describing the subset of data, said input comprising one or more data fields and, for each data field, respective field values from at least two sample web pages of a web page family for the web site; and

for each data field:

comparing respective web page code defining the respective field values for commonality to find a matching pattern with which to locate the respective field values, said matching pattern comprising a signature for the data field; and

defining the transcoding instructions in accordance with the matching pattern to locate and extract field values for the data field within web pages of the web page family.

2. The method of claim 1 wherein comparing respective web page code defining the respective field values comprises locating the respective field values in the respective web page code.

3. The method of claim 2 wherein comparing respective web page code defining the respective field values comprises locating object tags within the web page code.

4. The method of claim 3 further comprising constructing a programmatic data structure representing a hierarchy of object tags within the web page code and reviewing the hierarchy to determine the commonality.

5. The method of claim 1 wherein comparing respective web page code comprises performing pattern recognition to define a common pattern within the web page code with which to locate the respective field values.

6. The method of claim 1 wherein the web page code comprises markup language in plain

text.

7. The method of claim 8 wherein each signature comprises characters selected from the plain text of the web page code.

8. The method of claim 1 further comprising automatically defining the input in accordance with a tagging tool that identifies the respective field values from the sample web pages.

9. The method of claim 1 wherein the web site comprises an e-commerce web site for making a purchase.

10. The method of claim 1 further comprising defining transcoding instructions to express the extracted subset of data in a target format thereby to transcode the web page.

11. A system for automatically generating transcoding instructions to locate and extract a subset of data from a selected web page of a web site, the system comprising a processor and memory coupled thereto, said memory storing instructions and data to configure the processor for:

receiving an input describing the subset of data, said input comprising one or more data fields and, for each data field, respective field values from at least two sample web pages of a web page family for the web site; and

for each data field:

comparing respective web page code defining the respective field values for commonality to find a matching pattern with which to locate the respective field values, said matching pattern comprising a signature for the data field; and

defining the transcoding instructions in accordance with the matching pattern to locate and extract field values for the data field within web pages of the web page family.

12. The system of claim 11 wherein comparing respective web page code defining the respective field values comprises locating the respective field values in the respective web

page code.

13. The system of claim 12 wherein comparing respective web page code defining the respective field values comprises locating object tags within the web page code.

14. The system of claim 13 further comprising constructing a programmatic data structure representing a hierarchy of object tags within the web page code and reviewing the hierarchy to determine the commonality.

15. The system of claim 11 wherein comparing the web page code comprises performing pattern recognition to define a common pattern within the web page code with which to locate the respective field values.

16. The system of claim 11 wherein the web page code comprises markup language in plain text.

17. The system of claim 16 wherein each signature comprises characters selected from the plain text of the web page code.

18. The system of claim 11 further comprising automatically defining the input in accordance with a tagging tool that identifies the respective field values from the sample web pages.

19. The system of claim 11 wherein the web site comprises an e-commerce web site for making a purchase.

20. The system of claim 11 further comprising defining transcoding instructions to express the extracted subset of data in a target format thereby to transcode the web page.

21. A computer program product for automatically generating transcoding instructions to locate and extract a subset of data from a selected web page of a web site, the computer program product storing computer readable instructions which when executed by a computer processor configure the processor to:

receive an input describing the subset of data, said input comprising one or more

data fields and, for each data field, respective field values from at least two sample web pages of a web page family for the web site; and

for each data field:

compare respective web page code defining the respective field values for commonality to find a matching pattern with which to locate the respective field values, said matching pattern comprising a signature for the data field; and

define the transcoding instructions in accordance with the matching pattern to locate and extract field values for the data field within web pages of the web page family.

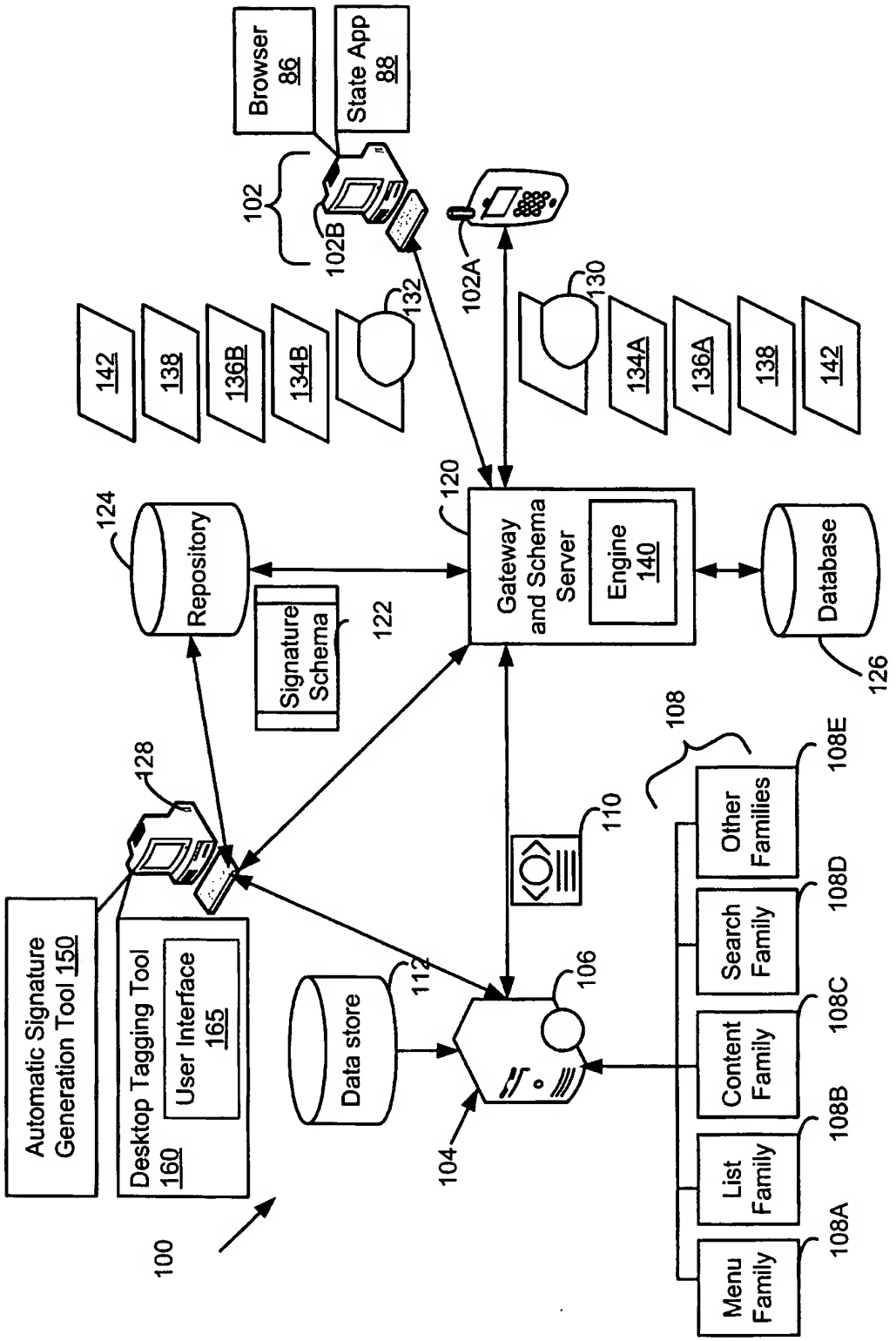
Browser 86
State App 88
102
102B
102A
132
130
142
138
136B
134B
134A
136A
138
142
124
Repository
120
Gateway and Schema Server
Engine 140
Database
126
Signature Schema
122
128
110
108
Other Families
108E
Search Family
108D
Content Family
108C
List Family
108B
Menu Family
108A
Automatic Signature Generation Tool 150
Desktop Tagging Tool 160
User Interface 165
Data store
112
106
104
100

Figure 1

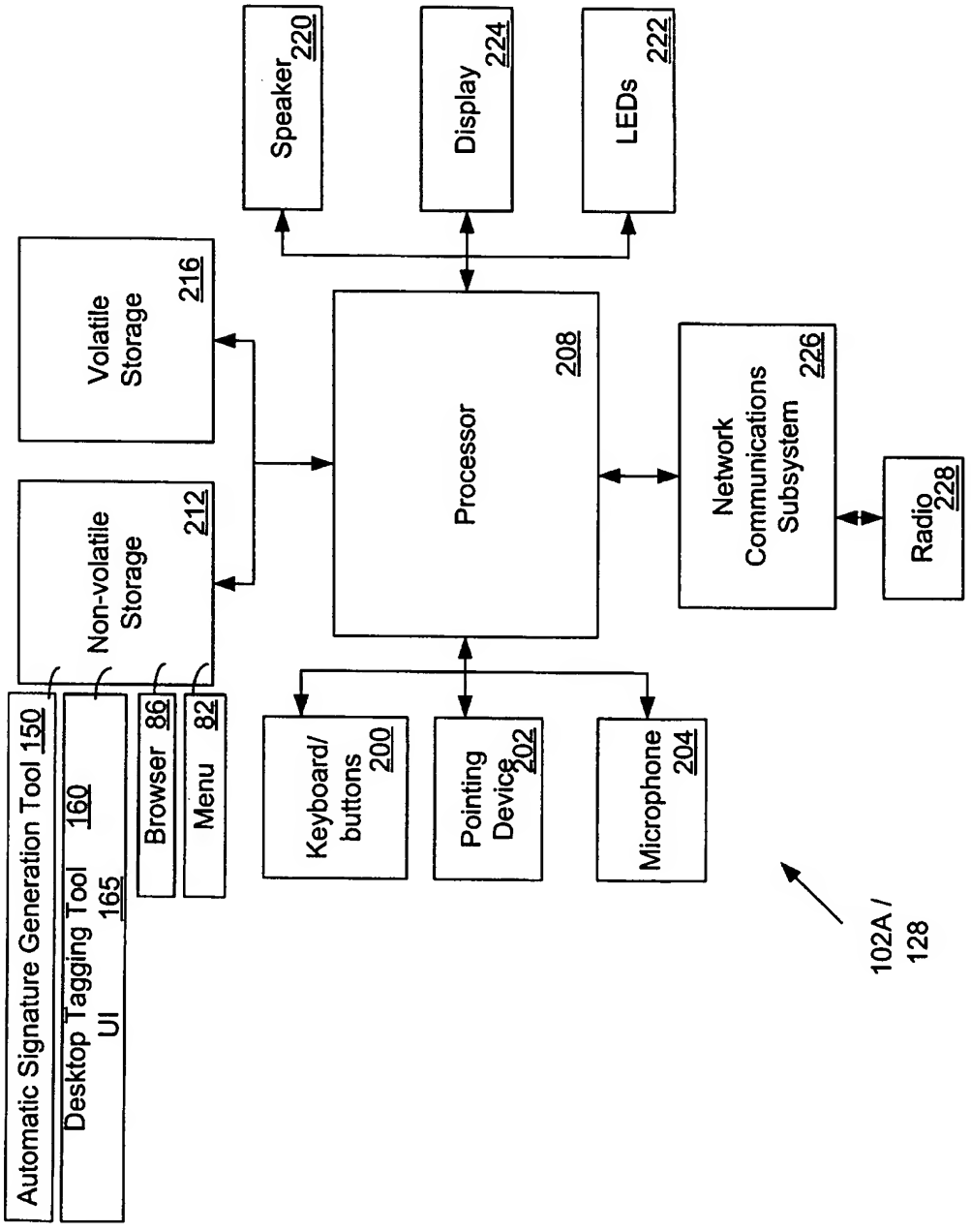
Speaker 220
Display 224
LEDs 222
Volatile Storage 216
Processor 208
Network Communications Subsystem 226
Radio 228
Non-volatile Storage 212
Keyboard/ buttons 200
Pointing Device 202
Microphone 204
Automatic Signature Generation Tool 150
Desktop Tagging Tool UI 160 165
Browser 86
Menu 82
102A / 128

Figure 2

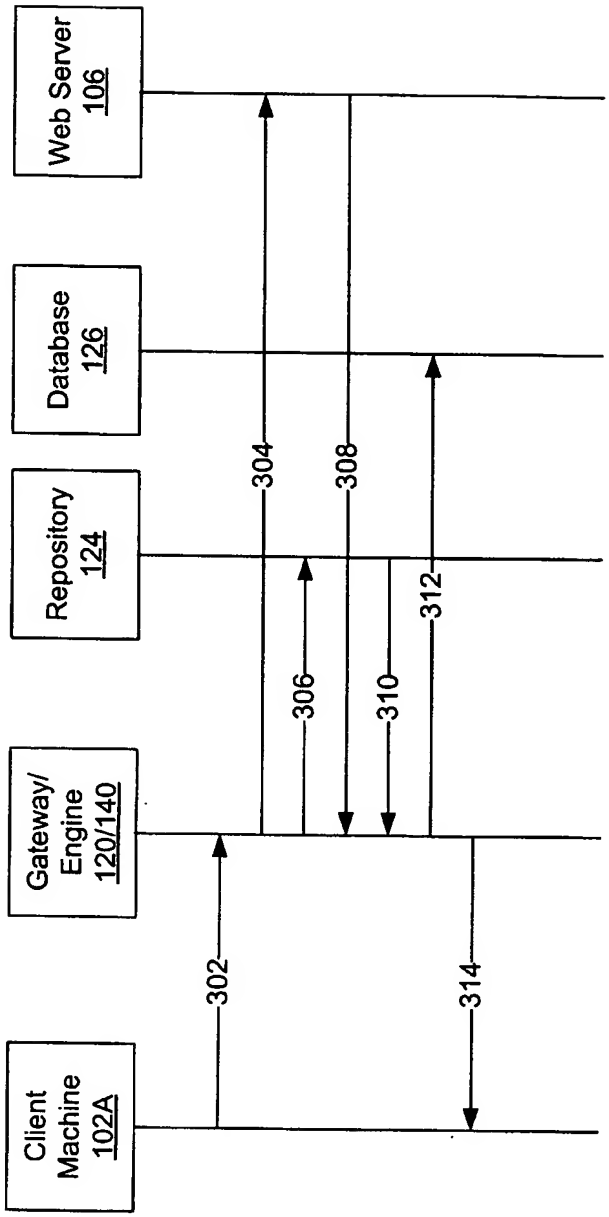
Web Server 106
Database 126
Repository 124
Gateway/ Engine 120/140
Client Machine 102A
302
304
306
308
310
312
314

Figure 3

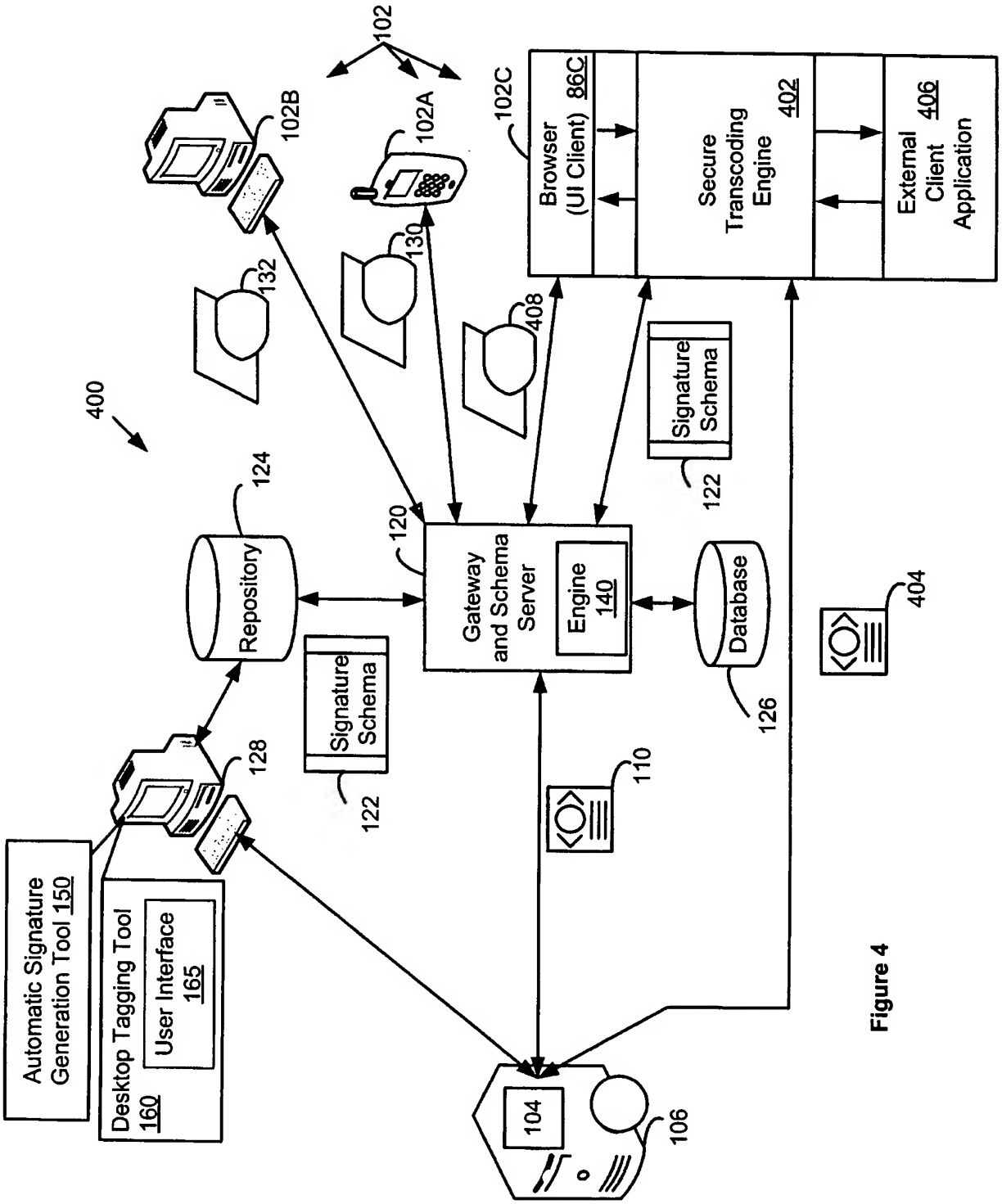
400
102
102A
102B
102C
Browser (UI Client) 86C
Secure Transcoding Engine 402
External Client Application 406
130
132
408
Signature Schema
122
124
Repository
120
Gateway and Schema Server
Engine 140
Database
126
404
110
128
Automatic Signature Generation Tool 150
Desktop Tagging Tool 160
User Interface 165
104
106

Figure 4

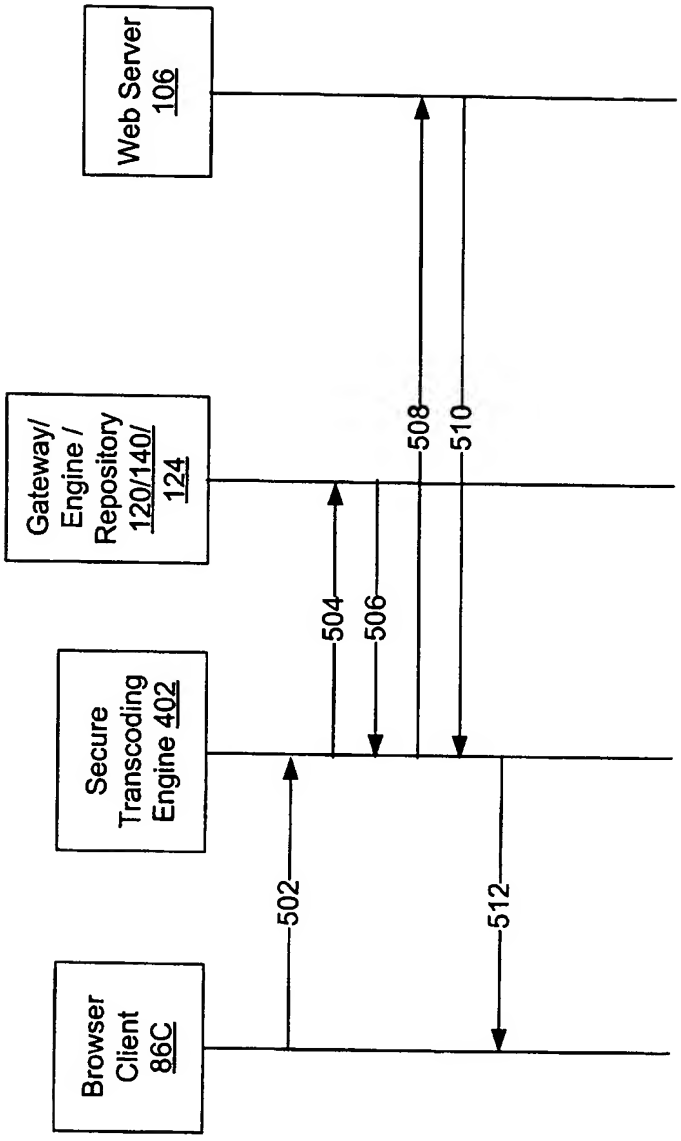
Web Server 106
Gateway/ Engine / Repository 120/140/ 124
Secure Transcoding Engine 402
Browser Client 86C
502
504
506
508
510
512

Figure 5

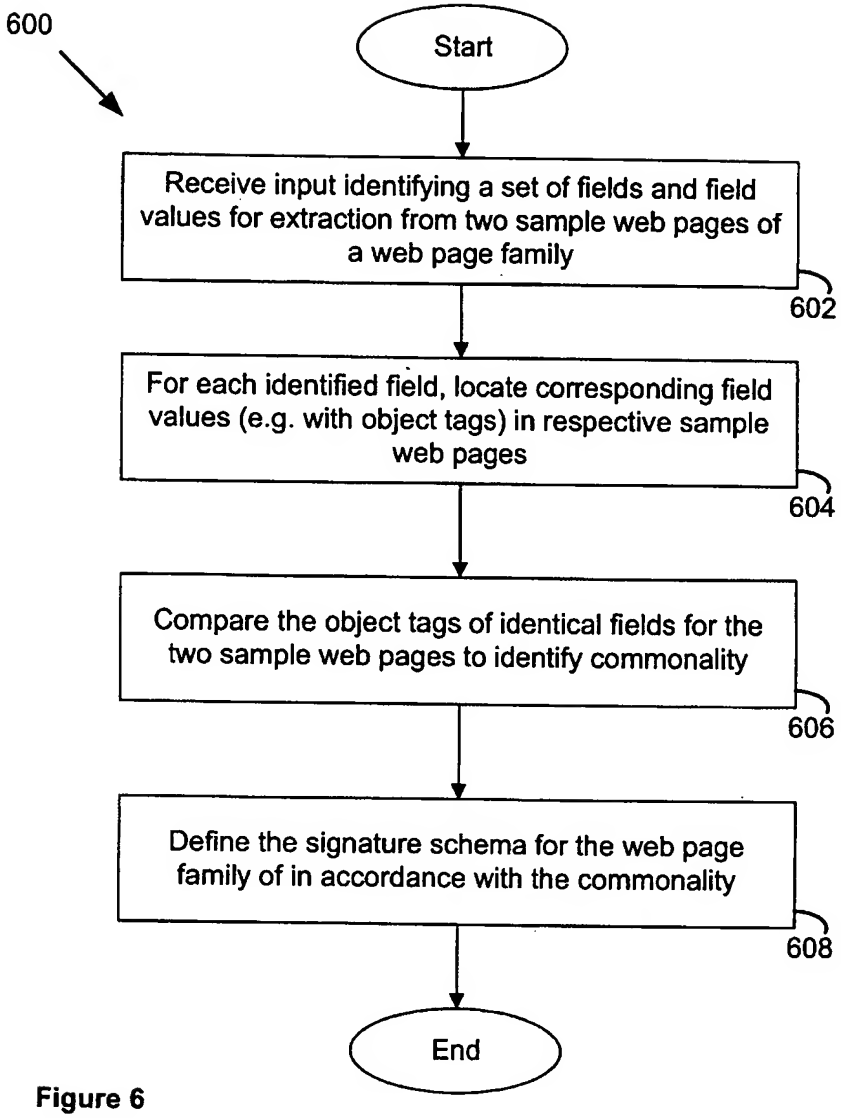
600
Start
Receive input identifying a set of fields and field values for extraction from two sample web pages of a web page family
602
For each identified field, locate corresponding field values (e.g. with object tags) in respective sample web pages
604
Compare the object tags of identical fields for the two sample web pages to identify commonality
606
Define the signature schema for the web page family of in accordance with the commonality
608
End

Figure 6

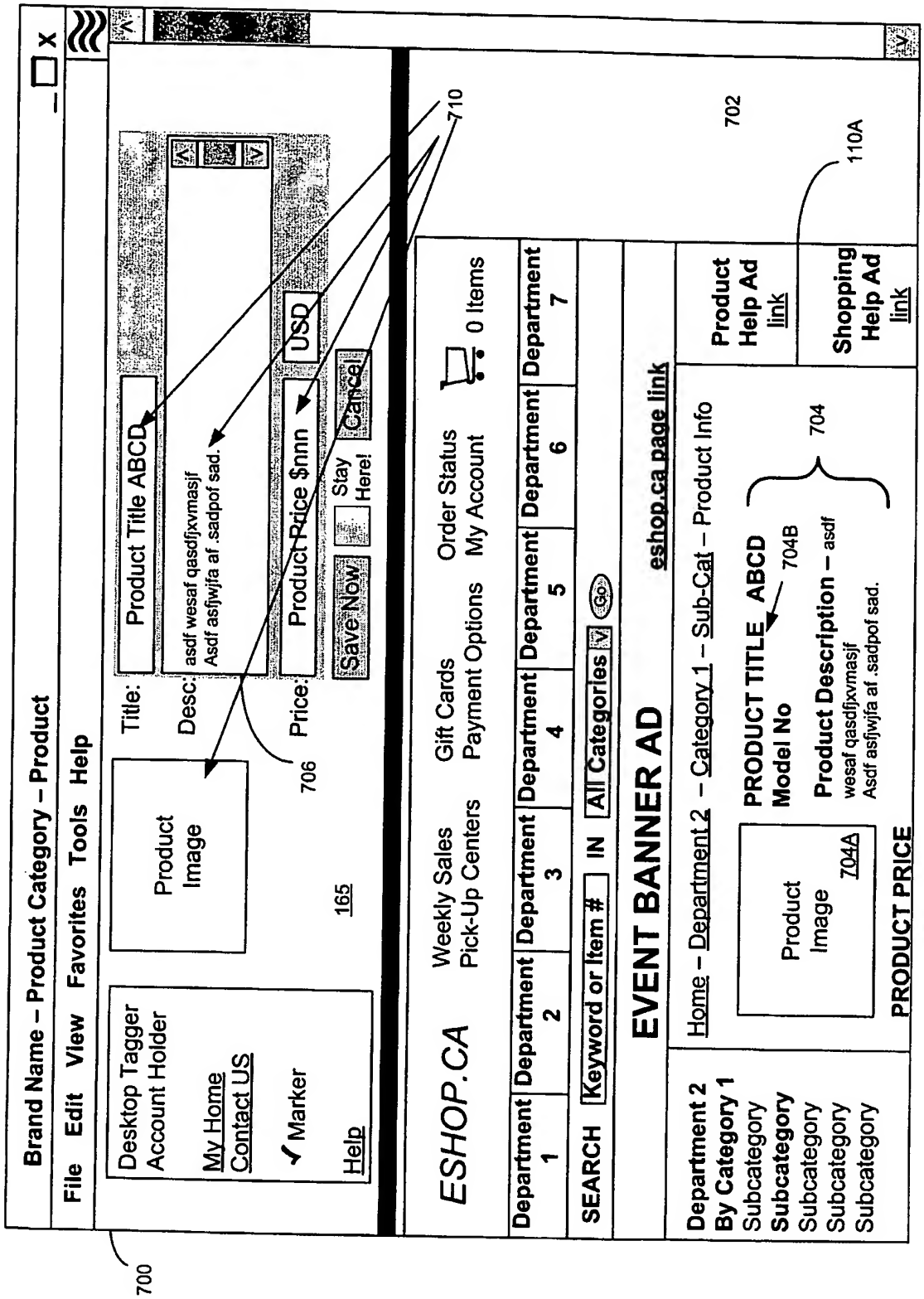
Brand Name – Product Category – Product
File Edit View Favorites Tools Help
Desktop Tagger
Account Holder
My Home
Contact US
Marker
Help
Product Image
Title:
Product Title ABCD
Desc:
asdf wesaf qasdfjxvmasjf
Asdf asfjwjfa af .sadpof sad.
Price:
Product Price $nnn
USD
Save Now
Stay Here!
Cancel
706
165
700
710
ESHOP.CA
Weekly Sales
Pick-Up Centers
Gift Cards
Payment Options
Order Status
My Account
0 Items
Department 1
Department 2
Department 3
Department 4
Department 5
Department 6
Department 7
SEARCH
Keyword or Item #
IN
All Categories
Go
EVENT BANNER AD
eshop.ca page link
702
Department 2
By Category 1
Subcategory
Subcategory
Subcategory
Subcategory
Home – Department 2 – Category 1 – Sub-Cat – Product Info
Product Image
704A
PRODUCT TITLE ABCD
Model No
704B
Product Description – asdf
wesaf qasdfjxvmasjf
Asdf asfjwjfa af .sadpof sad.
704
PRODUCT PRICE
Product Help Ad link
Shopping Help Ad link
110A

Figure 7

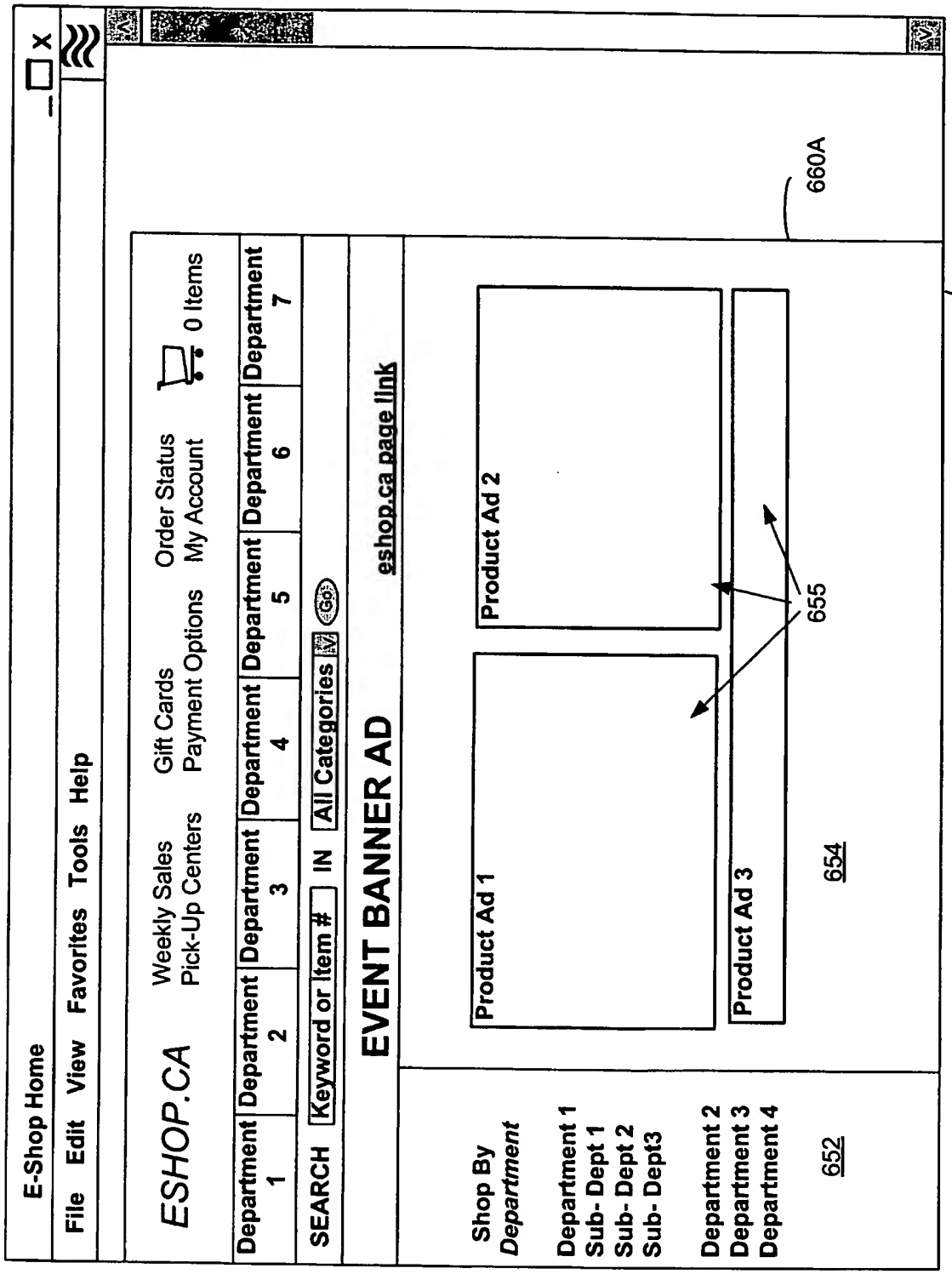
E-Shop Home
File Edit View Favorites Tools Help
ESHOP.CA
Weekly Sales
Pick-Up Centers
Gift Cards
Payment Options
Order Status
My Account
0 Items
Department 1
Department 2
Department 3
Department 4
Department 5
Department 6
Department 7
SEARCH
Keyword or Item #
IN
All Categories
Go
EVENT BANNER AD
eshop.ca page link
Shop By Department
Department 1
Sub- Dept 1
Sub- Dept 2
Sub- Dept3
Department 2
Department 3
Department 4
652
Product Ad 1
Product Ad 2
Product Ad 3
654
655
660A
650

Figure 8A

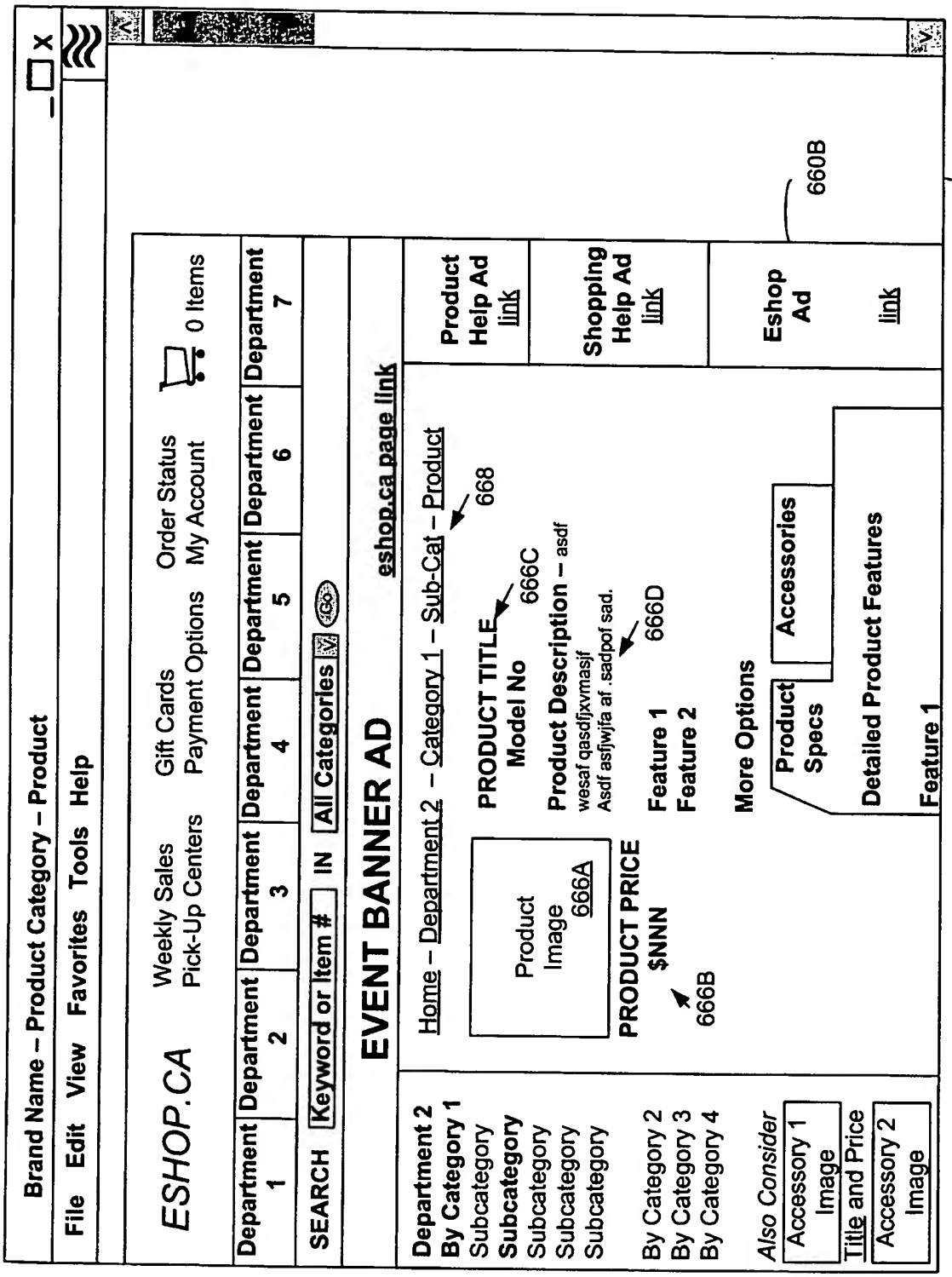
Brand Name – Product Category – Product
File Edit View Favorites Tools Help
ESHOP.CA
Weekly Sales
Pick-Up Centers
Gift Cards
Payment Options
Order Status
My Account
0 Items
Department 1
Department 2
Department 3
Department 4
Department 5
Department 6
Department 7
SEARCH
Keyword or Item #
IN
All Categories
Go
EVENT BANNER AD
eshop.ca page link
Department 2
By Category 1
Subcategory
Subcategory
Subcategory
Subcategory
By Category 2
By Category 3
By Category 4
Also Consider
Accessory 1 Image
Title and Price
Accessory 2 Image
Home – Department 2 – Category 1 – Sub-Cat – Product
668
Product Image
666A
PRODUCT TITLE
Model No
666C
Product Description – asdf
wesaf qasdfjxvmasjf
Asdf asfjwjfa af .sadpof sad.
PRODUCT PRICE
$NNN
666B
Feature 1
666D
Feature 2
More Options
Product Specs
Accessories
Detailed Product Features
Feature 1
Product Help Ad link
Shopping Help Ad link
Eshop Ad
660B
link
650

Figure 8B

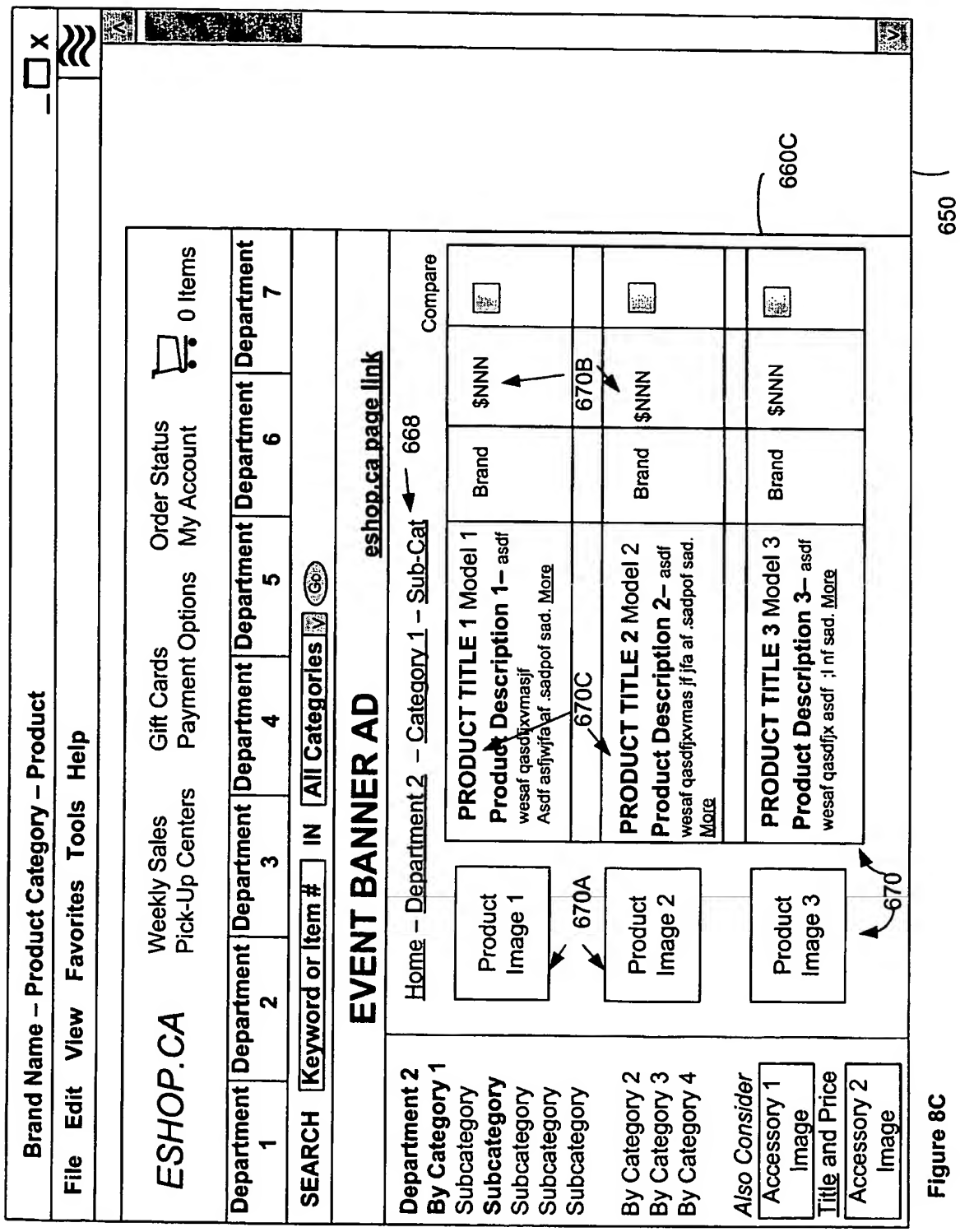
Brand Name – Product Category – Product
File Edit View Favorites Tools Help
ESHOP.CA
Weekly Sales
Pick-Up Centers
Gift Cards
Payment Options
Order Status
My Account
0 Items
Department 1
Department 2
Department 3
Department 4
Department 5
Department 6
Department 7
SEARCH Keyword or Item # IN All Categories Go
EVENT BANNER AD
eshop.ca page link
Department 2
By Category 1
Subcategory
Subcategory
Subcategory
Subcategory
By Category 2
By Category 3
By Category 4
Also Consider
Accessory 1 Image
Title and Price
Accessory 2 Image
Home – Department 2 – Category 1 – Sub-Cat
668
Compare
Product Image 1
Product Image 2
Product Image 3
670A
670
670C
PRODUCT TITLE 1 Model 1
Product Description 1– asdf
wesaf qasdfjxvmasjf
Asdf asfjwjfa af .sadpof sad. More
Brand
$NNN
670B
PRODUCT TITLE 2 Model 2
Product Description 2– asdf
wesaf qasdfjxvmas jf jfa af .sadpof sad.
More
Brand
$NNN
PRODUCT TITLE 3 Model 3
Product Description 3– asdf
wesaf qasdfjx asdf ;l nf sad. More
Brand
$NNN
660C
650

Figure 8C

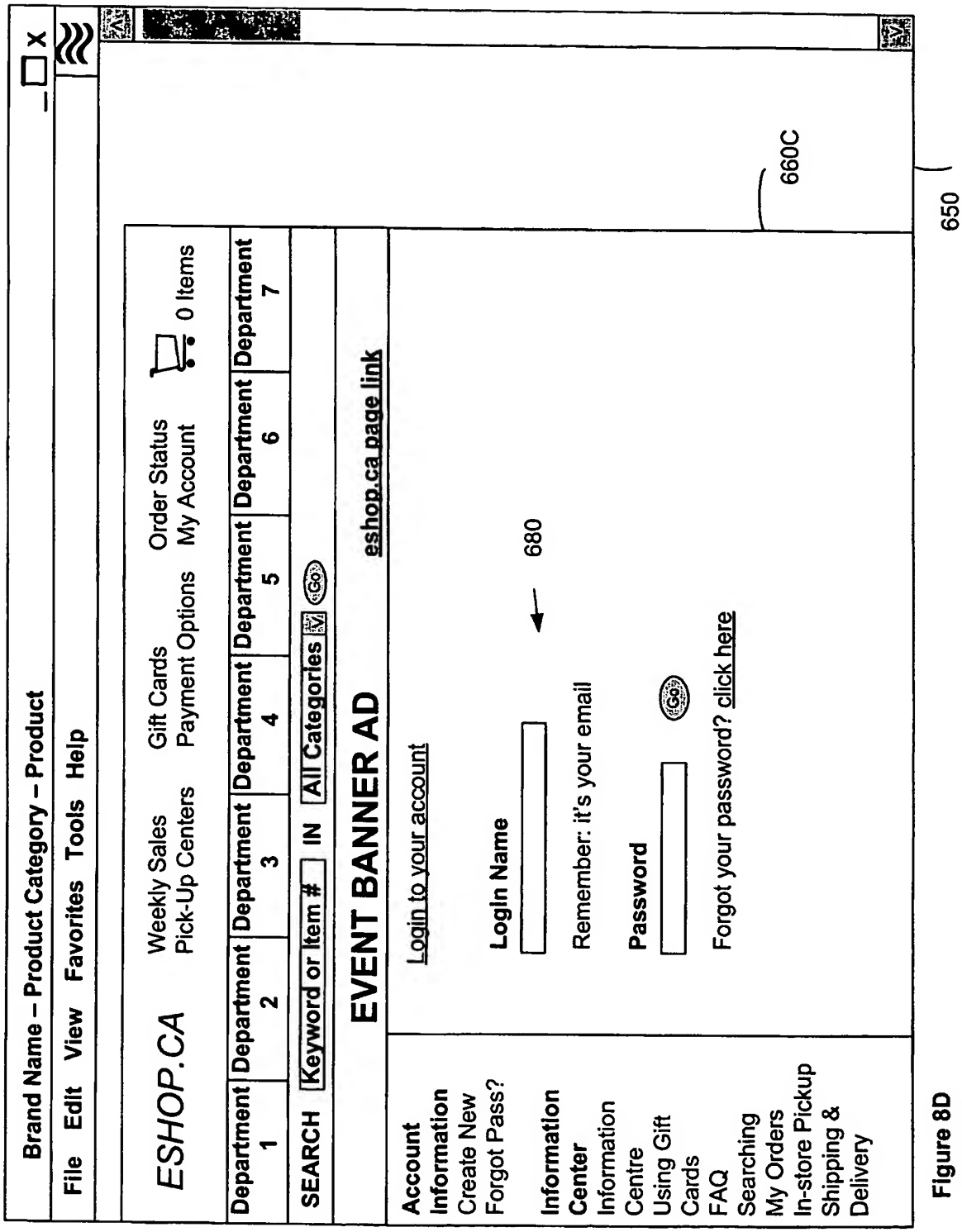
Brand Name – Product Category – Product
File Edit View Favorites Tools Help
ESHOP.CA
Weekly Sales
Pick-Up Centers
Gift Cards
Payment Options
Order Status
My Account
0 Items
Department 1
Department 2
Department 3
Department 4
Department 5
Department 6
Department 7
SEARCH
Keyword or Item #
IN
All Categories
Go
EVENT BANNER AD
eshop.ca page link
Account Information
Create New
Forgot Pass?
Information Center
Information Centre
Using Gift Cards
FAQ
Searching
My Orders
In-store Pickup
Shipping & Delivery
Login to your account
Login Name
680
Remember: it's your email
Password
Go
Forgot your password? click here
660C
650

Figure 8D

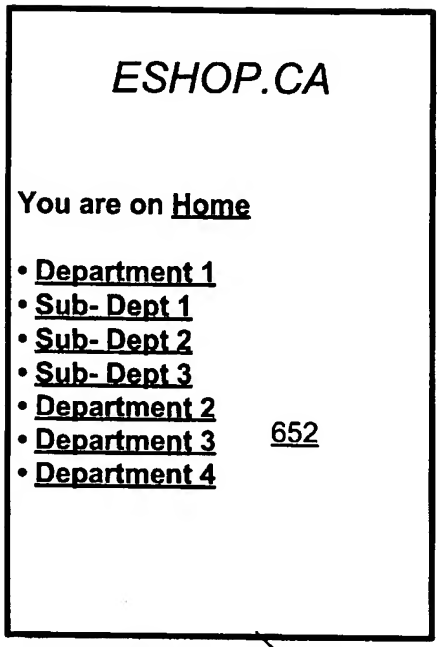

Figure 9A

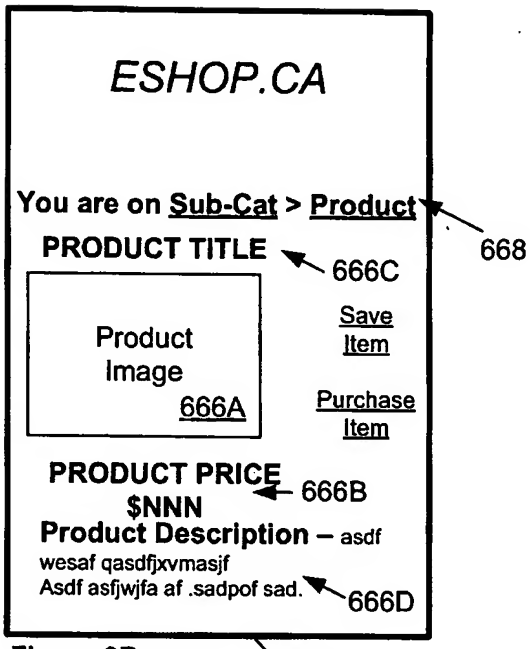

Figure 9B

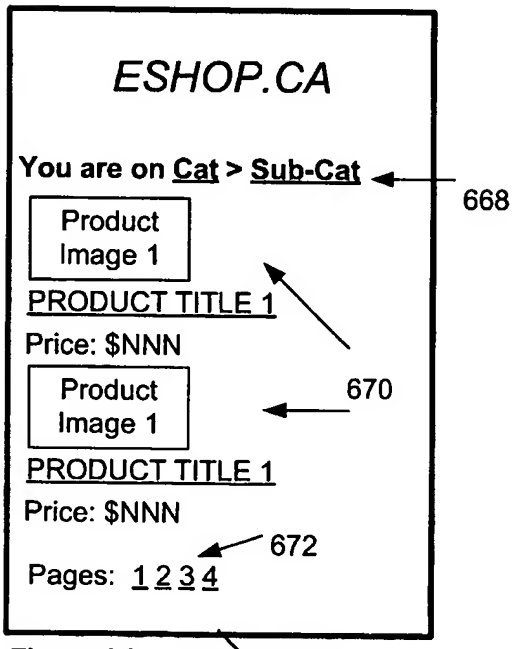

Figure 9C

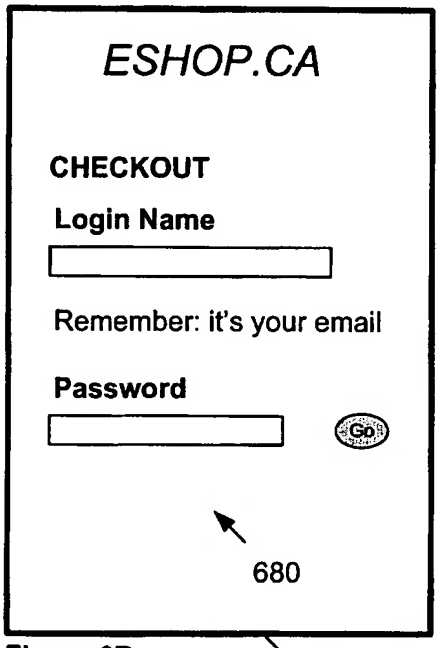

Figure 9D

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CA2008/000909

A. CLASSIFICATION OF SUBJECT MATTER
IPC: ***H04L 12/16*** (2006.01) , ***G06F 17/00*** (2006.01) , ***G06Q 30/00*** (2006.01) , ***H04Q 7/22*** (2006.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC: ***H04****, ***G06**** (2006.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)
Canadian Patent Database, United States Patent and Trademark Database, European Worldwide Database, Delphion, QPat and IEEE Xplore - Search terms used: transcod*, instruction, code, web page, subset, fields, values, (pattern or template or rule), match*, compar*, family, website, extract*

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2007/0038643 A1 (EPSTEIN) 15 February 2007 (15.02.2007) Whole document	1-21
A	US 2007/0033521 A1 (SULL et al.) 08 February 2007 (08.02.2007) Whole document	1-21
A	US 7,120,702 (HUANG et al.) 10 October 2006 (10.10.2006) Whole document	1-21
A	US 2005/0273772 A1 (MATSAKIS et al.) 08 December 2005 (08.12.2005) Whole document	1-21
A	KR 2004/0038458 A (SHIN HEE et al.) 08 May 2004 (08.05.2004) Whole document	1-21
A	US 2004/0078362 A1 (KIM et al.) 22 April 2004 (22.04.2004) Whole document	1-21

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents :
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
17 July 2008 (17-07-2008)	26 August 2008 (26-08-2008)
Name and mailing address of the ISA/CA Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001-819-953-2476	Authorized officer Donald Lefebvre 819- 997-2822

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2003/0018668 A1 (BRITTON et al.) 23 January 2003 (23.01.2003) Whole document	1-21
A	US 2002/0054090 A1 (SILVA et al.) 09 May 2002 (09.05.2002) Whole document	1-21
A	US 2002/0003547 A1 (WANG et al.) 10 January 2002 (10.01.2002) Whole document	1-21
A	EP 0 811 939 A2 (MIGHDOLL et al.) 10 December 1997 (10.12.1997) Whole document	1-21
P,A	JUNG-LEE, Hsiao et al., "Versatile transcoding proxy for internet content adaptation", IEEE Transaction on Multimedia, Volume 10, Issue 4, June 2008, pages 646-658. Whole document	1-21

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CA2008/000909

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
US2007038643A1	15-02-2007	AU2006278225A1	15-02-2007
		EP1934807A2	25-06-2008
		WO2007019571A2	15-02-2007
		WO2007019571A3	15-11-2007
US2007033521A1	08-02-2007	AU8300401A	05-02-2002
		KR20040074623A	25-08-2004
		KR20050002681A	10-01-2005
		KR20060043390A	15-05-2006
		KR20060096362A	11-09-2006
		KR20060099413A	19-09-2006
		KR20070028253A	12-03-2007
		KR20070101826A	17-10-2007
		KR20070103728A	24-10-2007
		KR20070111413A	21-11-2007
		US2002069218A1	06-06-2002
		US2003177503A1	18-09-2003
		US2004125124A1	01-07-2004
		US2004126021A1	01-07-2004
		US2004128317A1	01-07-2004
		US2005193408A1	01-09-2005
		US2005193425A1	01-09-2005
		US2005203927A1	15-09-2005
		US2005204385A1	15-09-2005
		US2005210145A1	22-09-2005
		US2006064716A1	23-03-2006
		US2007033170A1	08-02-2007
		US2007033292A1	08-02-2007
		US2007033515A1	08-02-2007
		US2007033533A1	08-02-2007
		US2007038612A1	15-02-2007
		US2007044010A1	22-02-2007
		WO0208948A2	31-01-2002
		WO0208948A3	25-09-2003
US7120702B2	10-10-2006	US2002133569A1	19-09-2002
US2005273772A1	08-12-2005	AU2286601A	03-07-2001
		CA2394058A1	28-06-2001
		EP1242907A2	25-09-2002
		JP2003518291T	03-06-2003
		US6772413B2	03-08-2004
		US7287217B2	23-10-2007
		US7318194B2	08-01-2008
		US2001056504A1	27-12-2001
		US2006235868A1	19-10-2006
		US2006236224A1	19-10-2006
		US2006236225A1	19-10-2006
		US2006253465A1	09-11-2006
		US2008040657A1	14-02-2008
		WO0146837A2	28-06-2001
		WO0146837A3	02-05-2002
		WO2005082102A2	09-09-2005
		WO2005082102A3	15-03-2007

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
KR20040038458	08-05-2004	AU2003274798A1	20040525
		CN1732459A	20060208
		CN100389415C	20080521
		EP1634183A1	20060315
		US2006230100A1	20061012
		WO2004040467A1	20040513
US2004078362A1	22-04-2004	KR20040034861A	29-04-2004
US2003018668A1	23-01-2003	None	
US2002054090A1	09-05-2002	None	
US2002003547A1	10-01-2002	AU6524201A	15-05-2002
		JP2001331407A	30-11-2001
		JP2002229843A	16-08-2002
		US2002007379A1	17-01-2002
		US2002174147A1	21-11-2002
		WO0237310A2	10-05-2002
		WO0237310A3	20-11-2003
EP0811939A2	10-12-1997	AT296446T	15-06-2005
		AU3139197A	05-01-1998
		AU3227597A	05-01-1998
		AU3375197A	05-01-1998
		AU5261298A	10-06-1998
		AU5446398A	29-06-1998
		AU5508898A	10-06-1998
		AU6026698A	07-08-1998
		AU6961498A	30-10-1998
		AU7684598A	10-02-1999
		CA2278023A1	23-07-1998
		DE69734080D1	06-10-2005
		DE69734080T2	14-06-2006
		DE69735463D1	11-05-2006
		DE69735463T2	02-11-2006
		DE69736373D1	07-09-2006
		DE69736373T2	23-08-2007
		DE69738443D1	21-02-2008
		DE69830301D1	30-06-2005
		DE69830301T2	02-02-2006
		DK960335T3	20-06-2005
		EP0811939A3	30-12-1998
		EP0811939B1	31-08-2005
		EP0811940A2	10-12-1997
		EP0811940A3	30-12-1998
		EP0811940B1	26-07-2006
		EP0812096A2	10-12-1997
		EP0812096A3	01-12-1999
		EP0812096B1	15-03-2006
		EP0844572A1	27-05-1998
		EP0844788A2	27-05-1998
		EP0844788A3	15-09-1999
		EP0844788B1	09-01-2008
		EP0848341A2	17-06-1998

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
EP0811939A2 (Continued)		EP0960335A2	01-12-1999
		EP0960335B1	25-05-2005
		EP0983665A2	08-03-2000
		EP0983665A4	11-05-2005
		EP0996892A1	03-05-2000
		EP0996892A4	20-04-2005
		EP1662747A2	31-05-2006
		EP1662748A2	31-05-2006
		EP1662749A2	31-05-2006
		EP1693769A2	23-08-2006
		EP1693769A3	20-09-2006
		ES2242995T3	16-11-2005
		JP3858346B2	13-12-2006
		JP4079288B2	23-04-2008
		JP10155039A	09-06-1998
		JP10171842A	26-06-1998
		JP10177372A	30-06-1998
		JP10187408A	21-07-1998
		JP10198571A	31-07-1998
		JP10228437A	25-08-1998
		JP2001519067T	16-10-2001
		JP2002512685T	23-04-2002
		JP2008108280A	08-05-2008
		JP2008117405A	22-05-2008
		KR100274135B1	15-12-2000
		KR100274738B1	15-12-2000
		KR100274739B1	15-12-2000
		USRE38915E1	06-12-2005
		USRE39866E1	02-10-2007
		US5830918A	03-11-1998
		US5851988A	22-12-1998
		US5896444A	20-04-1999
		US5918013A	29-06-1999
		US5935207A	10-08-1999
		US5940074A	17-08-1999
		US5945991A	31-08-1999
		US5974461A	26-10-1999
		US5996022A	30-11-1999
		US6005563A	21-12-1999
		US6008836A	28-12-1999
		US6023268A	08-02-2000
		US6034689A	07-03-2000
		US6073168A	06-06-2000
		US6133913A	17-10-2000
		US6141693A	31-10-2000
		US6230319B1	08-05-2001
		US6259442B1	10-07-2001
		US6278773B1	21-08-2001
		US6308221B1	23-10-2001
		US6308222B1	23-10-2001
		US6311197B2	30-10-2001
		US6311207B1	30-10-2001
		US6329431B1	11-12-2001
		US6330606B1	11-12-2001
		US6332157B1	18-12-2001
		US6473099B1	29-10-2002
		US6496205B1	17-12-2002
		US6496868B2	17-12-2002
		US6505232B1	07-01-2003
		US6584506B1	24-06-2003

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
EP0811939A2 (Continued)		US6614890B2	02-09-2003
		US6647421B1	11-11-2003
		US6662218B2	09-12-2003
		US6891553B2	10-05-2005
		US6957260B1	18-10-2005
		US7305472B2	04-12-2007
		US7350155B2	25-03-2008
		US2001003823A1	14-06-2001
		US2002013812A1	31-01-2002
		US2002016367A1	07-02-2002
		US2002021308A1	21-02-2002
		US2002048354A1	25-04-2002
		US2002054069A1	09-05-2002
		US2003014499A1	16-01-2003
		US2003078188A1	24-04-2003
		US2005149878A1	07-07-2005
		US2005188086A1	25-08-2005
		US2008141120A1	12-06-2008
		WO9746943A1	11-12-1997
		WO9747124A1	11-12-1997
		WO9747143A2	11-12-1997
		WO9747143A3	14-05-1998
		WO9822889A1	28-05-1998
		WO9823059A2	28-05-1998
		WO9823059A3	10-12-1998
		WO9825398A2	11-06-1998
		WO9825398A3	20-08-1998
		WO9832017A2	23-07-1998
		WO9832017A3	25-02-1999
		WO9845978A2	15-10-1998
		WO9845978A3	03-12-1998
		WO9904342A1	28-01-1999